SpaceFox

Zeste de savoir

L'internationalisation et ses pièges : créer un logiciel universel

10 décembre 2021

Table des matières

Introduction

C'est une excellente idée de rendre un programme, quel qu'il soit, accessible à un public différent de ses seuls développeurs. Et si l'internationalisation est une tâche à la réputation rébarbative, elle recèle en réalité quantité de détails passionnants… et piégeux. Ceci explique pourquoi tant de logiciels, sites, applications – y compris très connus – ont des défauts dans ce domaine.

Or, vois-tu, cher lecteur, chère lectrice, depuis une dizaine d'années maintenant je travaille sur des projets qui présentent une problématique forte d'internationalisation. C'est pourquoi je te propose un tour d'horizon des difficultés et subtilités à prendre en compte si toi aussi tu souhaites rendre ton programme accessible pour un maximum d'êtres humains.

On commencera par un peu de vocabulaire, quelques définitions pour que ce dont il est question soit bien clair. Puis viendront les pièges linguistiques, les différences culturelles, et enfin une poignée de considérations techniques.

Ce tutoriel ne nécessite pas de prérequis autres que celui de s'intéresser à l'internationalisation de logiciels.

1. Un peu de vocabulaire…

Introduction

Lorsqu'on veut traduire son logiciel, on se retrouve face à toute une foultitude de termes barbares à rallonge. Cette section va tenter d'éclaircir tout ça, histoire de savoir:

— où va le présent texte;
— ce qu'il ne couvre pas;
— quelles sont les clés pour rechercher des informations complémentaires.

1.1. L'internationalisation, ou « i18n »

En informatique, **l'internationalisation est le fait de rendre son logiciel apte à gérer plusieurs langues et cultures**. C'est une tâche principalement technique, faite par les équipes de développement – et c'est d'elle qu'il sera question dans ce tutoriel.

On trouve aussi très souvent le terme «**i18n**» parce qu'il y a 18 lettres entre le `i` et le `n` dans le mot anglais *internationalization*.

Le vocable **i18n** est **le** mot-clé à retenir pour retrouver les documentations au sujet de l'internationalisation qui s'appliquent à votre technologie préférée!

1.2. La régionalisation, ou « l10n »

En informatique, **la régionalisation** est l'action de traduire et adapter un logiciel pour une langue et une culture données. C'est une tâche principalement linguistique et culturelle, faite normalement par des traducteurs [1]footnote:1; et c'est ici qu'est gérée toute la problématique d'équilibrage entre traduction et adaptation. La régionalisation d'un programme impose que celui-ci soit internationalisé.

On trouve souvent le terme «**l10n**», parce qu'il y a 10 lettres entre le `l` et le `n` dans le mot anglais *localization*. Les francophones utilisent aussi le terme *localisation*, mais à tort, car c'est un faux-ami.

Le présent tutoriel ne couvre **pas** la régionalisation – principalement parce que je n'en ai pas les compétences. Si quelqu'un est motivé par un contenu sur le sujet…

1.3. Les autres termes

Si les deux vocables présentés sont très courants, on en retrouve d'autres, surtout chez de gros et anciens éditeurs qui ont mis en place ces technologies avant qu'elles ne soient très répandues. Voici quelques points d'entrée à tester si une recherche ne donne pas les résultats escomptés:

— IBM et Sun utilisent le mot *globalization*, qui regroupe les deux concepts ci-dessus.
— HP parle de *NLS*.
— Microsoft réunit les deux notions sous le seul terme *internationalization*.

1.4. Quand et quoi internationaliser ?

Internationaliser un programme est une tâche plutôt longue et complexe, en particulier:

Il ne suffit **pas** de **traduire** un logiciel pour **l'internationaliser**; la traduction n'est que la partie émergée de l'iceberg.

Subséquemment, une application qui permet la traduction de son interface, mais seulement cette traduction, sans considération supplémentaire, **ne permet pas** une localisation correcte.

D'autre part – et ce sera développé dans la section sur la technique –, l'internationalisation peut avoir des conséquences lourdes sur l'architecture du logiciel. Donc, internationaliser un programme déjà existant est une tâche longue, pénible, et qui peut conduire à des modifications bien plus profondes que simplement *ajouter une fonction de traduction à chaque fois qu'on affiche un texte*.

Tant que c'est possible, il vaut mieux se demander dès le début du projet si celui-ci doit être internationalisé, et le cas échéant dans quelles limites.

Conclusion

En résumé, ce tutoriel va s'intéresser à ce qu'implique de rendre son logiciel apte à gérer plusieurs langues et cultures, d'un point de vue technique et sans rentrer dans les considérations linguistiques. Voyons maintenant ce que veut dire concrètement «l'i18n».

1. [2]footnote:1 Et hélas, trop communément effectuée par les développeurs eux-mêmes, ce qui donne rarement de bons résultats… voire des bricolages à base de traduction automatique.

2. Gérer différentes langues

Introduction

Lorsqu'on parle d'internationalisation d'un logiciel, la première chose qui vient à l'esprit est de lui permettre d'être multi-langues. Donc, attaquons cette partie.

2.1. La sélection de la langue

La sélection de la langue, c'est *le* détail superimportant, qui a l'air simplissime… et qui est complètement raté dans la majorité des cas, y compris par les plus grands. Donc, s'il n'y a qu'une seule chose que vous devez retenir de ce tutoriel, c'est ça:

1. L'utilisateur peut **toujours** choisir la langue qu'il veut parmi toutes celles disponibles.
2. Le nom d'une langue est **toujours** dans la langue elle-même, jamais dans celle affichée.
3. On utilise **jamais** de drapeau pour représenter une langue.

C'est le moment de détailler ces points, dont chacun d'eux peut rendre l'internationalisation de votre application inopérante.

2.1.1. L'utilisateur doit pouvoir choisir la langue d'affichage

Ça a l'air idiot, mais on ne compte plus les logiciels qui, *bien que multilingues*, forcent l'affichage dans une langue sur un critère imposé par le développeur. Le problème est surtout vrai pour les sites internet, qui par nature peuvent être accessibles de partout dans le monde.

Or, je peux très bien visiter une page web depuis la France, dans un navigateur réglé en français… et ne pas lire le français. Que le site me propose cette langue par défaut c'est une excellente idée, mais il doit me laisser le choix.

2.1.2. Le nom de la langue est dans la langue elle-même

Ici rien que de la simple logique, mais à laquelle il faut penser: dans le sélecteur de langues, et contrairement à tout le reste de l'interface, on ne s'adresse pas à quelqu'un *qui parle la langue déjà sélectionnée* (et qui aurait besoin de comprendre la liste complète des langues) mais *à quelqu'un qui recherche une langue qu'il maîtrise dans la liste* (et donc qui a besoin de comprendre lesquelles sont disponibles.

C'est pourquoi les éléments de cette liste ne doivent jamais être traduits, et chaque nom de langue doit être donné dans la langue en question.

Un francophone a de grandes chances de sélectionner «french» dans une liste affichée intégralement en anglais; mais combien comprendront qu'ils doivent choisir «», «», «», «Ranskan kieli», « » ou encore « »?

Donc, si votre logiciel est disponible en français, anglais, russe et coréen, montrez «français, English, , » dans le sélecteur de langues, et ce quelle que soit la langue courante de l'application.

2.1.3. Un drapeau ne représente pas une langue

Un drapeau représente un état, une nation, une région… mais jamais une langue. Il y a un site entier sur le sujet (en anglais).

Ça n'est pas très naturel pour les Français qui comprennent français, parce que leur pays et leur langue partagent le même nom. Mais les contrexemples sont très nombreux dans les deux sens:

— Beaucoup de langues sont parlées dans plusieurs pays: le français est parlé en France, en Suisse, en Belgique, au Québec, en Afrique – d'où viennent beaucoup de lecteurs de Zeste de Savoir… Représenter le français par le drapeau de la France, c'est ignorer tous ces pays.
— Beaucoup de pays ont plusieurs langues officielles: la Belgique, la Suisse, le Canada… même l'Inde reconnait le français comme l'une de ses 22 langues régionales. Quelle langue désigne ces drapeaux?
— Dans le cas d'une langue qui tient son nom d'un pays, le pays d'origine n'est pas toujours celui qui a le plus de locuteurs de cette langue. Devrait-on évoquer l'anglais par le drapeau d'Angleterre (que personne ne connait), du Royaume-Uni (qui est la source historique) ou des USA (qui a le plus d'usagers)? L'espagnol doit-il être représenté par un drapeau d'Espagne ou du Mexique? Le portugais par un drapeau du Portugal ou du Brésil?
— Il y a des langues qui ne sont pas associées à un pays en particulier: quel drapeau représente l'arabe littéral, celui qui est généralement écrit?
— Quid des émigrés?

Donc, arrêtez de mettre des drapeaux pour désigner des langues. C'est au mieux une erreur, au pire une insulte.

Par contre, le drapeau reste un excellent moyen de sélectionner un pays ou une région, si par exemple certaines parties ne sont disponibles que dans une région donnée, ou que l'application gère les variantes locales d'une même langue (comme le français de France et du Canada).

(Promis, les sections suivantes sont plus courtes)

2.2. Différentes tailles de texte

Différentes langues, c'est aussi différentes tailles pour un même texte. Ça a un impact sur les éléments de design et la mise en page qui doivent être fluides pour éviter de se retrouver avec du texte qui déborde ou des composants presque vides.

On pense généralement à la **longueur** – l'anglais et l'italien sont d'ordinaire plus concis que le français qui lui-même est plus court que l'allemand.

On oublie plus souvent l'adaptation en **hauteur** nécessaire aux langues à alphasyllabaires ou à idéogrammes. Gérer ces langues (principalement asiatiques), c'est prendre en compte des textes beaucoup plus brefs, mais avec des lignes beaucoup plus hautes que leurs équivalents occidentaux. L'impact est surtout visible sur les éléments d'interface.

2.3. Le sens d'écriture

Toutes les langues ne s'écrivent pas de gauche à droite. Or, **le sens de lecture de l'interface utilisateur est le même que celui du langage**.

> *i* Lorsqu'on change de sens de langue, toute l'interface devrait s'inverser.

Pour mieux comprendre ce dont il s'agit, je vous laisse comparer la page d'accueil de Wikipedia en français et en arabe ; on y voit que tous les élément d'interface qui sont à gauche dans un idiome sont à droite dans l'autre, et vice-versa:

Notez que selon la technologie employée par votre logiciel, cette inversion peut être prévue et très facile (Android raisonnablement récent) ou au contraire horrible à réaliser (sites web sans framework qui prends déjà ce point en compte).

2.4. Tout ce qui vient avec une langue mais qui n'était pas prévu

Ah oui, parce qu'on n'y pense pas forcément, mais chaque langue vient avec ses petites spécificités qui peuvent casser des points que le développeur a considérés comme acquis et évidents.

2.4.1. Le cas du pluriel

Certains éléments peuvent s'afficher au singulier ou au pluriel selon les circonstances. Non seulement il faut prendre ce fait en considération; mais surtout, les règles d'accord dépendent du langage.

Si le français fait usage du singulier et du pluriel (deux cas) d'autres langues ont des systèmes de nombres bien plus complets et complexes . Dans les cas «fréquents», on trouve une gestion spécifique du duel (deux éléments) et du paucal (quelques éléments).

L'accord de l'absence d'éléments dépend de la langue: certaines utilisent le singulier; l'anglais le pluriel; le français le concept flottant de *«on met le nombre qui devrait être utilisé s'il y avait un nombre normal du truc absent»* [1]footnote:1 (bonne chance !).

1. [2]footnote:1 Typiquement en alimentation: *sans sel* (il y a normalement *du sel* au singulier), mais *sans sucre**s*** (il y a normalement *des sucres* alimentaires, au pluriel). Et encore, cette règle est discutée…

2.4.2. Généralisation : le cas des accords, déclinaisons et autres spécificités

Plus généralement, toute clé de traduction à trous (du style `{personnage} ramasse {objet}`) peut se transformer en prise de tête et en gestion de cas particuliers. Parce que des cas spéciaux à gérer peuvent arriver:

— Si le genre du personnage impose un accord plus loin dans la phrase – c'est fréquent en français.
— Si le genre de l'objet impose une conjugaison propre – c'est fréquent en italien.
— Si l'objet impose une déclinaison spécifique selon le type d'objet – c'est fréquent en allemand, ou dans les langues extrême-orientales avec la notion de classificateur .
— Et on pourrait multiplier les exemples à l'infini.

Ici, pas de miracle: il faut trouver un équilibre entre «chaque variante a une clé de traduction spécifique» (ce qui donne des combinatoires ingérables) et «la traduction doit se débrouiller avec les clés qu'on lui fournit» (ce qui peut contraindre à des phrases vraiment pas naturelles). L'équilibre est en plus différent selon le type d'application: un *framework* technique se contentera très bien de traductions à la hache, tandis qu'un jeu vidéo narratif aura besoin d'un respect soigné des ambiances. C'est l'occasion pour l'équipe de développement de discuter avec celle de traduction, c'est souvent très enrichissant!

2.4.3. Gestion de la casse

Il se peut qu'un bout de votre application change la casse de certains textes. Déjà, la distinction entre majuscule et minuscules n'existe pas dans toutes les langues, et toutes les langues n'utilisent pas les mêmes règles sur ce qu'est la «bonne» casse[3]footnote:2

Pire encore: changer la casse d'un mot peut donner un résultat différent selon la langue! Je vous laisse imaginer les dégâts si vous comptiez sur cette modification d'un point de vue technique.

L'exemple typique est celui du `I` turc . Certaines langues (à commencer par le turc) ont **deux** lettres `i`: avec et sans point `i` et , qui se capitalisent respectivement en et `I`. Donc, si la langue courante est une de ces langues, `id` va devenir `İD`, ou `ID` va devenir `ıd` après un changement de casse… alors que le développeur attendait respectivement `ID` ou `id`.

2.4.4. Gestion des diacritiques

Ceci nous amène à la problématique plus générale des diacritiques(ces symboles qu'on rajoute aux lettres, comme les accents). Le principe est simple:

> On ne supprime **jamais** les diacritiques. Il n'y a plus de raison de faire ça[5]footnote:3.

2. [4]footnote:2 L'allemand colle des majuscules aux noms communs. L'anglais américain fait des titres avec une majuscule à chaque mot, ce qui ne se fait pas du tout en français. Etc.
3. [6]footnote:3 …sauf si vous devez vous interfacer avec un antique système. Mais là c'est l'occasion de 1. vérifier que ce système ne peut pas être mis à jour et 2. de faire un projet dédié, avec étude d'impacts, et tout.

En français, on arrive généralement à saisir le sens d'un texte privé de ses accents, et les lettres accentuées ne sont pas considérées comme différentes de leurs variantes nues; mais c'est *très* loin d'être le cas dans toutes les langues.

Quant aux notions de *slug* et autres bizarreries qu'on trouve par exemple dans les URL: vous avez un souci d'encodage et non de diacritiques. Wikipedia est l'un des plus gros sites au monde et emploie les signes de toutes les langues dans ses URL – vérifiez par vous-même.

Si votre crainte (légitime) est qu'un utilisateur abuse de votre largesse en employant des caractères différents, mais visuellement identiques (comme les `Α` grec, `А` cyrillique et `A` latins), ce n'est pas un problème de diacritique, mais d'homoglyphes et Unicode fournit les outils pour les résoudre.

2.4.5. Ordre alphabétique

L'ordre alphabétique dépend de l'alphabet utilisé (y compris au sein d'un même groupe d'alphabets, par exemple tous les alphabets «latins»). C'est à prendre en compte si vous ne voulez pas perdre vos usagers.

Conclusion

Ainsi donc, gérer les langues dans son logiciel, c'est bien plus que permettre d'afficher des traductions. Ça nécessite:

— Un sélecteur de langues, avec chaque nom de langue dans sa langue elle-même et non représenté par un drapeau.
— Une ergonomie qui accepte des textes de diverses tailles, en longueur comme en hauteur.
— Une organisation de l'espace qui suit le sens de lecture de la langue choisie.
— … et plein de détails spécifiques à chaque langue, comme les pluriels, les accords, la casse, l'ordre alphabétique et j'en passe.

Il est maintenant temps de s'attaquer aux différences culturelles – car oui, elles ont aussi un impact technique.

3. Les différences culturelles

Introduction

L'internationalisation, ce n'est pas *que* gérer différentes langues. C'est aussi gérer différentes cultures – et ce au sein *d'une même langue*. Ce point est le plus important de tous, car il est souvent oublié alors que c'est là qu'on trouve les différences les plus complexes à gérer et les plus handicapantes si elles sont oubliées.

3.1. Les noms et civilité des utilisateurs

La façon de nommer les personnes est très, très variable selon les cultures.

3.1.1. Les notions de « nom » et « prénom »

Les simples notions de *prénom* et *nom de famille* n'existent pas dans toutes les cultures, loin de là – on peut lire à ce sujet les articles sur le nom personnel et le nom de famille ainsi que les contenus liés. Même quand ces deux noms sont usités, leur ordre n'est pas fixe – doit-on mettre le nom personnel avant ou après le nom de famille?

Ne changez pas les noms de vos usagers

N'oubliez jamais: **vos utilisateurs savent comment ils s'appellent, pas vous**. Ils peuvent légitimement:

- avoir une dénomination *très* longue;
- avoir une dénomination *très* courte;
- avoir une dénomination qui comporte des caractères très exotiques[1footnote:1].

3.1.2. Titres et prédicats

L'usage des titres et prédicats de civilité est lui aussi fluctuant. Si en France on se satisfait généralement de la paire «Monsieur/Madame», en Allemagne on appréciera une plus grande variété de titres. Il est fréquent que quelqu'un titulaire d'un doctorat y utilise le prédicat «Docteur», tandis que ce titre est presque réservé aux médecins en France.

1. [2]footnote:1 Rien qu'en France c'est problématique. Ainsi, beaucoup de «François» se trouvent condamnés à s'appeler «Francois»… ici ça reste compréhensible, mais dans d'autres cas ça porte à confusion.

Rien de tout ça ne doit donc être codé en dur dans l'application pour que celle-ci soit internationalisable…

Quelques exemples

Tout ceci est d'autant plus vrai que votre logiciel a besoin des noms *formels* des usagers.

- Les noms arabes peuvent devenir très longs, comme
- De même pour le système traditionnel espagnol: le nom complet de Felipe de Marichalar y Borbón est en réalité *Felipe Juan Froilán de Todos los Santos de Marichalar y Borbón*
- Hery Rajaonarimampianina a pour nom de naissance *Hery Martial Rajaonarimampianina Rakotoarimanana*
- Inversement dans le cas de «U Nu», «U» est le titre, et «Nu» le nom complet de cette personne.
- Les Russes utilisent très peu les prédicats, votre application devrait pouvoir gérer´´´´ si l'actuel président doit s'y inscrire.
- Et je ne parle pas des *middle name* américains, des systèmes du Sri Lanka ou d'Islande, etc.

3.1.3. Et donc comment faire ?

La seule solution simple et fiable est d'employer un champ unique pour tout ça, avec une invite qui dit quelque chose comme *«Comment souhaitez-vous que l'on vous appelle?»*.

3.2. Les adresses postales

L'adresse peut être le seul point d'internationalisation à gérer – dans le cas d'une application monolangue, mais qui utilise des adresses dans plusieurs pays – et pourtant, c'est l'un des plus casse-pieds parce que les spécificités sont nombreuses.

i

Le format d'une adresse se gère **par pays**, indépendamment de la langue.
Mettez le **moins de contraintes** possibles sur une adresse.

Voici quelques-uns des éléments à prendre en compte si vous devez gérer des adresses internationales.

3.2.1. La longueur des adresses peut être *très* variable

Une adresse française d'un particulier fait généralement deux à trois lignes en plus du nom et du pays. Mais il peut y avoir des lignes internes à une entreprise.

Certains pays ont des systèmes d'adresses qui nécessitent plus de lignes – je pense aux adresses en Chine ou en Nouvelle-Calédonie, qui peuvent atteindre cinq lignes en plus du nom et du pays. En fait la notion de numéro de rue et même d'adresse dans une rue peut ne pas exister .

Donc, si le formulaire limite au triptyque «Adresse / Complément d'adresse / Code postal + Ville»… certains utilisateurs peuvent se retrouver bloqués! Wikipedia propose un échantillon de différents formats d'adresse à l'international .

3.2.2. Le code postal

Le code postal peut:

— Ne pas être un nombre
— Comporter entre 4 et 9 caractères, plus des séparateurs variables
— Être placé au début ou à la fin d'une ligne, voire seul au début de l'adresse
— Ne pas exister *du tout*

Autant dire que si dans votre base de données le code postal est un nombre de cinq chiffres, vous allez au-devant d'ennuis.

3.2.3. Les bases de données d'adresse

Les bases de données d'adresse ne sont par définition **jamais** à jour, à cause de la création de nouvelles voies, des renommages, des renumérotations, des cas particuliers… Vous pouvez vous en servir pour aider au remplissage d'une adresse, mais sauf obligation légale, n'imposez pas que l'utilisateur doive saisir une adresse connue.

Quelques exemples

3.2.3.1. Deux adresses françaises

```
Pierre Dupont
Dupond SA
Service comptable
52 RUE DES JONQUILLES
BP 77 BELLEVILLE
99123 VILLENOUVELLE
```

Notez la différence de longueur et le fait que la seconde adresse n'a ni voie ni numéro.

```
Pierre Dupont
Le Lieu-Dit
99987 Petit Village
```

3.2.3.2. Une adresse anglaise

Le code postal contient des lettres et est placé seul sur sa ligne.

```
Mr John SMITH
1 Vallance Road
Bethnal Green
LONDON
E2 1AA
```

3.2.3.3. Une adresse russe

Outre l'alphabet cyrillique, on peut avoir beaucoup de lignes.

```
Пьянков Андрей Сергеевич
ул. Ореховая, д. 25
пос. Лесное
Алексеевский р-н
Воронежская обл.
Россия
247112
```

3.2.3.4. Au Japon

Les systèmes d'adresse japonais (il y en a plusieurs) commencent par le code postal et autorisent une adresse (hors nom) sur une seule ligne; ici pour l'ambassade de France – l'une ou l'autre variante:

```
106-8514 4-11-44
4-11-44 Minami-Azabu, Minato-ku, Tōkyō 106-8514
```

3.2.4. … et donc, comment gérer une adresse ?

i

Idéalement une adresse c'est deux champs libres :

1. la dénomination complète de la personne (ou de la société),
2. l'adresse complète.

C'est le moyen le plus simple de gérer tous les cas possibles.

S'il existe des contraintes de vérification (légales, marketing…) n'oubliez pas que:

— Chaque pays a son format spécifique, donc ses règles de validation propres;

— Le système de stockage doit permettre de gérer tous ces différents cas;
— Certains éléments fréquents (numéro dans la rue, type de voie…) sont en réalité facultatifs.

3.3. Les nombres, les monnaies et tout ce qui va avec

Le formatage des nombres – sous forme numérique – dépend de la langue, et parfois de la culture au sein d'une même langue.

3.3.1. Le système numéral

Si beaucoup de langues utilisent les chiffres dits «arabes (occidentaux)», ce n'est pas systématique. D'ailleurs, beaucoup de pays arabes se servent des chiffres «arabes orientaux » différents des premiers…

3.3.2. Les séparateurs décimaux et « de milliers »

Le séparateur décimal d'un nombre varie selon la langue, tout comme le séparateur de milliers. L'article Wikipedia en anglais est assez complet sur la question.

D'ailleurs, le séparateur de milliers porte assez mal son nom, parce que dans certains cas les chiffres sont groupés par autre chose que par milliers: on trouve de groupements par 10 000 (groupes de 4 chiffres) ou des formes plus complexes, et des exceptions (1000 et non 1 000).

3.3.3. Le cas des monnaies

Pour afficher un prix, le logiciel doit prendre en compte toutes les contraintes d'affichage des nombres, et y ajouter:

— La gestion du symbole monétaire, qui peut être placé avant ou après, avec ou sans séparateur.
— L'affichage de la partie fractionnaire de la monnaie («centimes», mais ce n'est pas toujours des centièmes), qui peut être *très* différente[3] footnote:1 sur une même monnaie selon le pays.
— Le nombre de chiffres sur la partie fractionnaire peut différer selon la monnaie, avec des exceptions (2 chiffres pour l'euro, mais 3 chiffres pour certains produits comme l'essence au litre).
— Les règles légales d'arrondi dans les calculs et affichages qui peuvent varier.

D'ailleurs, si quelqu'un connait une référence des formats d'affichage des prix selon le pays, ça m'intéresse.

1. [4]footnote:1 Certains pays affichent la partie fractionnaire comme un nombre à virgule normal, d'autre ont des affichages plus exotiques comme une partie fractionnaire mise en exposant, ou un symbole monétaire qui sert de séparateur décimal. On peut donc avoir des écritures comme 12,34 €, 12.34€, 12^{34}, 13€34… Certains pays, comme la Belgique, utilisent aussi le tiret pour remplacer la partie fractionnaire dans un prix entier: 12,00 € deviendra 12.— €.

3.3.4. Taxes, paiement et frais de livraison

Puisqu'on parle de monnaies, les implémentations des systèmes de taxes, de frais de livraisons et des moyens de paiement sont très variées selon les pays.

— L'Amérique du Nord a un système de taxes atrocement compliqué[5]footnote:2.
— La Russie (et tous les très grands pays) doit gérer des livraisons complexes, sans prix connu à priori.
— Chaque pays a ses habitudes et ses prestataires en matière de règlement.
— Sans oublier que forcer le paiement sur un fournisseur unique peut engendrer des commissions importantes pour le consommateur international.

Est-ce qu'il y a pire comme bazar culturel à gérer que les prix? Peut-être bien…

3.4. Les calendriers, dates et heures

3.4.1. Calendrier et dates

Non seulement tous les pays n'utilisent pas le même calendrier , mais en plus, en considérant le seul calendrier grégorien:

— Chaque pays a plusieurs formes d'affichage des dates (courte en chiffres, longue avec le nom du jour et le mois en lettres, intermédiaires);
— Certains éléments (comme le nom des jours ou des mois) n'existent pas dans toutes les langues;
— Au sein d'une unique langue, le format peut changer (10/02/2019 en France, mais 10.02.2019 en Suisse);
— Au sein d'un même type de format et pour une même langue, l'ordre des composantes peut varier selon le pays[7]footnote:1;
— La norme officielle peut n'être usitée par personne dans la vie courante.[8]footnote:2

Même les données associées à un calendrier qui ne sont pas directement des dates sont variables!

— Le jour qui commence la semaine n'est pas toujours le même (lundi en France, dimanche aux USA)
— Le calcul du numéro de la semaine en cours est donc dépendant du pays

i

Pour l'affichage correct d'une date à l'utilisateur, on doit connaitre sa langue *et* son pays.
Wikipédia a un tableau assez complet sur le sujet
Pour la communication avec d'autres systèmes d'information, il faut impérativement employer une norme sous peine de graves erreurs.

2. [6]footnote:2 Un cliché veut qu'un système de commerce électronique en ligne américain gère très mal les langues et pays variés, mais très bien les taxes; tandis qu'un système européen gère très bien les langues et pays, mais très mal les taxes complexes.

1. [9]footnote:1 D'ailleurs le format bizarre mois/jour/année n'est pas employé par la langue anglaise, mais par les seuls USA et le Canada anglophone.

2. [10]footnote:2 La norme canadienne demande d'utiliser année-mois-jour, mais en réalité les francophones emploient jour/mois/année, et les anglophones mois/jour/année…

3.4.2. Quelques mots sur l'heure

Comme les dates, chaque langue et chaque pays a ses petites habitudes pour afficher l'heure. Mais surtout, la terre étant approximativement une sphère, il faut aussi gérer les fuseaux horaires… Sachant que rien ne garantit que plusieurs serveurs soient à la même heure, ni même que les différents outils au sein du même serveur soient sur le même fuseau horaire (c'est possible !).

En fait, le sujet des dates et des heures est si compliqué qu'il dispose maintenant de son tutoriel dédié, que je vous invite à consulter:

Dates, durées et horloges en informatique

3.5. Les systèmes d'unités

La théorie est simple: tous les pays du monde, sauf les USA, la Birmanie et le Libéria, utilisent le Système International d'unités .

Sauf que nous ne vivons pas – hélas – en Théorie.

En pratique, chaque territoire a ses petites habitudes d'usage d'unités hors SI, dans tout un tas de domaines impossibles à deviner (la liste très partielle sur Wikipedia est déjà édifiante), et *évidemment* sans aucune forme d'uniformisation au niveau mondial. C'est tellement vrai qu'on peut avoir une unité de même nom dans une même langue qui a deux valeurs différentes selon le pays, comme le gallon qui ne définit pas le même volume selon que le locuteur est anglais (4,546 litres) ou natif des USA (3,785 litres… ou 4,405 litres en fonction du cas !).

Donc, les unités de mesure sont à internationaliser sous peine d'erreurs monumentales .

3.6. Le design et les couleurs

On touche ici à quelque chose de plus profond et de plus difficile à gérer que tout ce qui a été présenté avant, parce que beaucoup plus impactant sur le logiciel dans son ensemble.

3.6.1. Le design

Selon la culture, l'utilisateur va avoir des attentes différentes sur ce qu'est un *bon design*: ce qui sera considéré comme acceptable ici sera mauvais ailleurs, et inversement. L'exemple classique nous vient d'Extrême-Orient, où les designs très chargés en texte semblent appréciés.

Voici la une d'un grand quotidien conservateur japonais et un quotidien français de même sensibilité :

3. Les différences culturelles

De même avec les unes d'un grand journal sportif japonais et de son pendant français :


日刊スポーツ
野球 MLB サッカー 海外サッカー スポーツ ゴルフ 相撲・格闘技 競馬 ボート競輪オート 芸能 社会
大坂なおみがバイン・コーチと関係解消
ファミマ店員を解雇 ナメた態度の不適切動画を撮影
日本ハム斎藤佑樹 2回無安打無失点「良かった」
ファンサービスで悲劇…松坂は開幕抹消、再び離脱も
香川真司「突き詰めたい」中盤の王様と代表復帰の道
「やっぱりPL出てる選手やな」／中村順司4
広島小園に沖縄切符！恩師前で無安打も走攻守で評価
改名！日向坂「ひゅうが」なのに何故「ひなた」
元フジの「住職」松倉悦郎アナ逮捕、鍵で車場つける
えっ…松倉容疑者モットーは「不条理に対する怒り」
ニュースランキング
記事 写真 Facebook
今日の占い
クローズアップ情報
海外サッカー
セリエA ブンデス プレミア スペイン 欧州CL
プロ野球
Jリーグ
新着コラム

C'est hélas difficile de véritablement jouer sur ce point tant les modifications et spécificités que ça imposerait sont importantes.

3.6.2. La signification des couleurs

Les couleurs ont une symbolique plus ou moins claire mais surtout très dépendante de la culture. Là aussi, c'est presque impossible à gérer en pratique; ce point existe surtout pour signaler un risque de contresens.

Par exemple, les bourses occidentales signalent les gains en vert (positif) et les pertes en rouge (négatif); mais les bourses extrême-orientales comme celle de Hong-kong ou Séoul , donnent les gains en rouge (faste) et les pertes en vert (néfaste).

3.7. Les contraintes politiques et légales

Internationaliser un logiciel, et le rendre officiellement disponible dans tel ou tel pays, c'est aussi devoir gérer tout un tas de contraintes politiques et légales. Là, il n'y a pas le choix, il faut se renseigner au cas par cas. Ce qui revient souvent, c'est:

— Les obligations et restrictions techniques: hébergement de données dans une zone précise, interdiction de technologies…

— La représentation de pays sur les cartes[11]footnote:1: la liste des territoires contestés est telle qu'il est impossible d'avoir une carte considérée comme vraie pour tous les pays; une carte est donc un élément à traduire *dans sa géographie*.
— La censure sous toutes ses formes.
— Les politiques de modération des contenus publiés par les utilisateurs.
— Les contraintes à la publication et à l'achat en ligne.
— …

Conclusion

Eh oui! Les différences culturelles ont de gros impacts sur l'application, sa technique et son ergonomie. Les points d'attention, non exhaustifs, sont les suivants:

— Soyez d'une grande tolérance dans la façon dont vos usagers se nomment, les désignations sont très variées de par le monde.
— Les adresses postales sont tout aussi diverses – et les bases d'adresses jamais à jour.
— Chaque culture a sa manière d'afficher un nombre, ses petites manies pour gérer les prix, et ses règles légales de paiement.
— Le calendrier grégorien n'est pas le seul existant, et même en son sein on trouve des différences d'utilisation notables.
— Le Système International d'Unités n'est hélas pas encore universel.
— Une «bonne ergonomie» est un concept culturel, ceci jusque dans le choix des couleurs.
— Dans tous les cas, n'oubliez jamais les piles de lois locales.

Évidemment, gérer autant de spécificités implique des contraintes purement techniques, qui sont l'objet de la partie suivante.

1. [12]footnote:1 La Mer du Japon s'appelle «Mer de l'Est» en Corée. Ça, c'est facile, ça se gère avec la traduction. Plus difficile: représenter le Cachemire comme intégralement en Inde est une obligation légale en Inde.

4. Quelques considérations techniques

Introduction

C'est bien beau de parler d'internationalisation, de mettre des grands mots et de pointer des pièges; mais techniquement ça implique quoi?

Eh bien… ça dépend énormément du type de logiciel et de la technologie utilisée. Néanmoins il existe un certain nombre de constantes que l'on va passer en revue.

4.1. Utilisez des outils pour faire le boulot à votre place !

C'est sans doute le conseil le plus important de cette section, alors je vous le remets en bien visible:

> Utilisez des outils pour faire le boulot à votre place!

Une grande partie de ce que je vous ai présenté est *déjà* géré par tout un tas de composants et de bibliothèques, voire parfois directement dans les standards [1] footnote:1. Employez ces outils au lieu de réinventer la roue, ça vous évitera de la concevoir carrée.

Mais si vous choisissez une bibliothèque tierce, assurez-vous qu'elle fasse le travail correctement.

4.2. Éléments traduits ou langues porteuses d'éléments ?

S'il y a *une* décision à prendre le plus tôt possible dans le développement du logiciel, c'est celle-ci:

> Est-ce que chaque élément de l'application sera traduit; ou bien est-ce que chaque langue sera porteuse d'éléments non traduits?
> Autrement dit, est-ce que le résultat sera une application strictement identique dans toutes les langues, ou est-ce que chaque langue aura des contenus spécifiques?

1. [2] footnote:1 Les sélecteurs de date en HTML gèrent beaucoup de spécificités de dates; les bibliothèques standards des bons langages de programmation gèrent les dates, les fuseaux horaires, les encodages; etc. Renseignez-vous!

L'interface de l'application est naturellement traduite dans toutes les langues – ou alors disponible dans une seule langue d'administration. La question se pose pour le contenu qui peut être importé ou contribué:

— Soit chaque élément (article, etc.) existe dans toutes les langues. La navigation est similaire dans l'application quelle que soit la langue, et chaque «page» de l'application est visible dans toutes les langues.
— Soit chaque langue a son propre ensemble d'éléments, et donc l'application peut présenter des données et des parcours utilisateurs différents selon la langue.

Ceci implique des choix techniques forts, d'où l'importance d'une décision rapide dans le processus de développement.

4.3. Clés de traduction et la langue « développeur »

Tôt ou tard, le développement de l'application nécessitera d'employer des clés de traduction, des chaines de caractère qui permettent de dire: «Ici, tu affiches ce texte dans la langue sélectionnée par l'utilisateur».

4.3.1. Clés signifiantes, clés techniques et mutualisation

Il faut choisir si les clés de traduction sont plutôt *signifiantes* (le texte qui sert de clé est lui-même humainement lisible, comme `Bouton de connexion`), ou plutôt *technique* (le texte est une représentation purement technique, comme `commons.button.login`).

L'outil d'internationalisation peut influencer vers l'une ou l'autre des solutions, sans qu'il n'y en ait de meilleure. Une clé de traduction signifiante pose moins de problèmes en cas de traduction manquante, mais rend la détection de tels oublis plus complexes.

L'autre question qui se pose est celle du découpage et du regroupement des clés entre elles. Généralement, la majorité des clés de traductions seront uniques et évidentes à associer (toutes celles qui concernent la page d'accueil, etc.) mais certaines se retrouvent un peu partout et donnent très envie d'être réutilisée.

C'est tout à fait possible si l'on respecte deux conditions:

1. Les clés réutilisables sont clairement identifiées comme telles, en particulier par l'équipe de traduction.
2. Toute modification de la valeur de la clé ou de son attribution implique une vérification de cohérence de tous les usages de la clé.

Sans quoi on se retrouve avec des blagues comme cet outil de virtualisation qui indique qu'un serveur est démarré depuis *«0 deuxièmes»*, parce que la même clé a servi à traduire *«second»*, la seconde comme unité de temps, et *«second»*, la deuxième place…

4.3.2. La langue « développeur »

Les développeurs ne sont ni des linguistes ni des ergonomes. Mais ce ne sont pas non plus des machines et ont besoin d'une version lisible du logiciel pendant l'élaboration, et donc d'avoir une *langue par défaut*.

Le plus simple est de les laisser rédiger la véritable langue par défaut du logiciel… mais c'est le meilleur moyen d'avoir une version bancale et pleine de fautes.

Une autre solution, c'est d'avoir une langue *spécifique aux développeurs*, et une version «propre» de cette même langue, destinée au public, dans les fichiers de traduction. Selon l'outil d'internationalisation utilisé, cette langue «pour les développeurs» peut même ne pas être exprimée comme les autres: certains outils permettent de donner une valeur par défaut si une clé de traduction est manquante – c'est fait naturellement si on emploie des clés signifiantes.

4.4. Unicode, votre meilleur ami

Nous sommes en 2019, mais on a encore des surprises, alors je préfère le préciser.

Oubliez tous les encodages de caractères locaux, tout votre texte devrait être en Unicode .

Sauf si le système impose autre chose, employez UTF-8 , éventuellement UTF-16 si votre application gère surtout des langues asiatiques.

Si vous devez vous interfacer avec d'autres systèmes informatiques qui n'utilisent pas un de ces encodages, assurez-vous que la conversion est faite le plus tôt possible pour ce qui est des entrées, et le plus tard possible en ce qui concerne les sorties. Et quoi qu'on vous dise, n'oubliez pas: **le «texte brut» n'existe pas** .

4.5. Le cas particulier des messages d'erreur techniques

Dans un logiciel, il y a deux choses qui ne doivent *en aucun cas* être internationalisées:

1. Les noms des langues dans le sélecteur de langue, comme déjà mentionnés plus tôt;
2. Les messages d'erreur *à destination des développeurs*.

On peut avoir l'impression d'aider le développeur en lui donnant un message d'erreur dans sa propre langue. C'est une bévue: dans l'immense majorité des cas, la première étape effectuée face à un message d'erreur dont la solution n'est pas triviale, c'est de le chercher sur Internet, pour vérifier s'il n'y a pas de la documentation à ce sujet, ou quelqu'un d'autre qui a une réponse.

Si le message est traduit, il y a un fort risque que la documentation ou l'aide ne soit pas disponible dans toutes les langues, et donc la personne qui développe va perdre du temps au lieu d'en gagner. Et si jamais ce type de message doit absolument être traduit – parce qu'il

peut être affiché à l'utilisateur final, par exemple – alors, assurez-vous de fournir aussi un **code d'erreur** non traduit.

4.6. Rendre un site internet accessible partout dans le monde

Si votre logiciel est un site internet que vous hébergez, il est *naturellement* disponible partout dans le monde.

Du moins, ça, c'est la théorie.

En pratique, il sera rapide dans les zones géographiquement proches d'où sont les serveurs sur lesquels il est déployé… et lent ailleurs, la faute aux lois de la physique. Or, la vitesse de chargement des pages n'est pas qu'un critère de confort pour l'utilisateur: un site trop mou peut décourager les visiteurs, ce qui peut être un gros problème – une perte de chiffre d'affaires pour un site de vente. Il faut donc choisir avec soin l'emplacement géographique de ses serveurs, *y compris en cas d'hébergement dans les nuages.*

De plus, si le service doit être accessible avec efficacité depuis plusieurs zones géographiques distinctes, des stratégies existent pour améliorer les temps d'accès: instances multiples CDN …

Conclusion

En résumé, les principales considérations techniques pour l'internationalisation sont:

1. D'utiliser des outils pour le faire – à commencer par Unicode.
2. De décider au plus tôt qui va porter les traductions dans l'application, et quel sera le format des clés de traduction.
3. Dans le cas d'un site ou d'une application connectée à Internet, de concevoir une infrastructure capable de délivrer une qualité de service correcte partout dans le monde.

N'oubliez pas que les messages d'erreurs ne devraient être traduits que s'ils contiennent un code d'erreur identique pour toutes les langues.

Il y aurait bien plus à en dire, mais hélas beaucoup sont spécifiques au langage de programmation voire aux logiciels utilisés…

Conclusion

Ainsi donc, rendre un logiciel utilisable à l'international est une tâche assez simple en apparence, mais qui recèle une multitude de pièges à éviter et de détails à gérer.

Ainsi, un logiciel correctement internationalisé:

— Verra ses différentes langues accessibles et sélectionnables par l'usager;
— Gèrera les subtilités inhérentes aux multiples langues qu'il supporte: différences de longueurs de textes, de hauteur de ligne, de sens de l'interface le cas échéant, ainsi que toutes les difficultés spécifiques à ces langues;
— Supportera les différences culturelles telles que la gestion des adresses, dates, nombres dont les monnaies… sans présenter d'éléments mésinterprétables par les utilisateurs des différents pays;
— Sera cohérent d'un point de vue technique sur sa façon de gérer l'internationalisation.

Merci à @informaticienzero pour la validation, ainsi qu'à @Fumble, @ache, @Renault, @Javier et @QuentinC pour leurs remarques et conseils avisés pendant la bêta de ce tutoriel!

Liste des abréviations

CDN Content Delivery Network. 26

NLS Native (ou National) Language Support. 5

SI Système International (d'unités). 18