Håndtering af tekstoversættelse i flersprogsapplikationer

Når man arbejder med internationalisering af systemer, er der en udfordring i hvordan man håndterer forskelligheden i sprogenes entals- og flertalsformer. Denne artikel forsøger at komme med et bud på, hvordan man kan håndtere dette.

Skrevet den **29. apr 2013** af **softspot** I kategorien **Programmering / Generelt** | ⭐⭐⭐⭐☆

Revisionshistorik:

```
25-08-2012: artikel oprettet.
25-04-2013: rettelse af engelsk mønster (tak til olebole), samt
            tilpasning af indledningsteksten.
29-04-2013: tilføjelse af link til konkret løsningsforslag i C#
```

Hvis man skal lave et site der kan håndtere forskellige sprog er der mange måder at gøre dette på.

En metode er at vedligeholde flere sites i de forskellige sprog man ønsker at understøtte. En fordel ved dette er, at man så kan styre sitets udseende 100% indenfor hvert sprog. Det samme kan dog også være en ulempe, når man skal lave den samme ændring på alle sprogversioner.

En anden metode kunne være at benytte samme sideskabeloner og blot indsætte tekst-pladsholdere fra en ordbog de steder, hvor indholdet skal sprogstyres. Fordelen er her, at man kun skal vedligeholde layout ét sted, men omvendt er differentiering af layoutet, afhængig af sprogvalget, mere komplekst.

Jeg vil i denne artikel beskrive en version af den sidste metode.

Når man vedligeholder en ordbog skal der ofte tages højde for en eller flere variable værdier i formuleringerne af teksterne i de forskellige sprog. Dette kunne f.eks. komme til udtryk i flg. tekst:

```
"Der findes 20 personer"
```

Dette ser lidt anderledes ud, hvis der kun findes 1 person

```
"Der findes 1 person"
```

Håndteringen af forskellen her, er til at overkomme, men man skal trods alt tage stilling til om der skal bruges flertals- eller entalsform for ordet "person". Dette kan gøres inline på siden:

```
personEndelse = ""
if antalPersoner <> 1 then
  personEndelse = "er"
end if
```

```
Translate("Der findes " & antalPersoner & " person" & personEndelse)
```

Translate er her den funktion, som tager teksten og finder en tilsvarende oversættelse i ordbogen til det aktuelt valgte sprog og returnerer den.

Der er umiddelbart tre udfordringer her:

1. Teksten indeholder en variabel (antalPersoner)

2. Placeringen af variablen er ikke nødvendigvis den samme på alle sprog

3. Teksten findes i 2 varianter, hvis man ser bort fra variablen

Den første og anden udfordring kan løses ved at indsætte en pladsholder for variablen i teksten og så sende værdien med i en parameter til Translate.

```
Translate("Der findes {0} person" & personEndelse, antalPersoner)
```

Dette giver dog kun mulighed for at sende én parameter med til oversætteren. Det er givetvis ikke nok i mere komplekse oversættelsesscenarier. Overvej f.eks. flg. tekst:

```
"Der findes X personer i Y registre"
```

hvor X er antal personer og Y er antal registre, altså to forskellige værdier.

Pladsholdere til værdier
Dette kan løses ved at sende et array af værdier med til Translate-funktionen.

```
Translate("Der findes {0} personer i {1} registre", array(antalPersoner, antalRegistre))
```

Som man kan måske kan se er udfordring 3 fra tidligere blevet endnu større, da der nu findes 4 varianter af den sidste tekst

1. ental af person
2. flertal af person
3. ental af register
4. flertal af register

Det begynder at blive noget omstændigt at vedligeholde disse tekster!

For at det ikke skal være løgn, kan der endda være forskel på hvormange former et ord har afhængig af sproget, så man kan altså ikke altid regne med at det er en eller flere der adskiller om der skal flertalsendelse på.

Der er altså brug for endnu mere fleksibilitet i formateringen af den oversatte tekst.

Mere fleksibilitet

Jeg har flg. forslag til hvordan man kan håndtere formendelser i tekster.

Der anvendes en form i den oversatte tekst, som tillader at angive ord eller endelser alt efter hvilken værdi der sendes med til funktionen. Det kunne se således ud:

```
"Der findes {0} person[#0!1:er] i {1} regist[#1!1:re|=1:er]"
```

Ovenstående er altså ikke tekst-pladsholderen, men den oversatte tekst som hentes frem via pladsholderen sammenholdt med sproget (som her er dansk). Pladsholderen kunne se således ud:

```
"Der findes X personer i Y registre"
```

Den engelske oversættelse kunne se nogenlunde således ud:

```
"{0} person[#0!1:s] exist[#0=1:s] in {1} register[#1!1:s]"
```

Som man kan se, er der ikke behov for de samme betingelser i den engelske oversættelse, som i den danske. Derfor er det vigtigt at forstå, at det er i OVERSÆTTELSEN betingelserne skal indsættes og ikke i tekst-pladsholderen. Tekst-pladsholderen er blot en nøgle til at få fat i oversættelsen med og kunne i princippet være et tal. Personligt foretrækker jeg bare, at tekst-pladsholderen giver mening i den kontekst hvor den skal bruges. Der ligger naturligvis nogle performance- og resurseovervejelser i formatet af tekst-pladsholderen, da lange tekstuelle tekst-pladsholdere givetvis kræver flere resurser (RAM, database, netværk osv.) at benytte.

Betinglser i oversættelserne
Formatet af betingelserne er således:

[#<feltnummer><betingelse>{|<betingelse>}]

```
<feltnummer>   := et tal fra 0 og opefter, som peger på en
                  af de medsendte værdier
<betingelse>   := <operator><værdi>:<indsæt tekst>
<operator>     := !|=|<|>
<værdi>        := et heltal
<indsæt tekst> := vilkårlig tekst der skal indsættes
                  hvis udtrykket er sandt
```

Dvs. værdien af feltnummer kan enten være

```
! : forskellig fra
= : lig med
< : mindre end
> : større end
```

den numeriske værdi og hvis betingelsen er opfyldt, så indsæt teksten efter kolon (og frem til næste pipe-karakter, dvs. lodret streg) og afslut evalueringen af det aktuelle udtryk.
Hver betingelse ud over den først foranstilles med en pipe-karakter.

Med ovenstående format kan vi specificere meget mere komplekse oversættelsesmønstre med meget færre oversættelser (dvs. 1 pr. tekst der skal oversættes pr. sprog). Det er umiddelbart en optimal situation!

Fleksibilitet for en pris
Denne teknik kommer dog med nogle omkostninger, nemlig at der skal ske en fortolkning af formatet hver gang oversættelsen til en tekst skal hentes. Der kan foretages caching i forskellige grader, men det ændrer ikke på at oversættelse af en tekst involverer et funktionskald og en fortolkning af hvilken betingelse der er gældende.
Det vigtigste er umiddelbart, at kompleksiteten vedr. det, at oversætte en tekst, er pakket ind i en funktion og dermed er oversættelseslogikken også lettere at optimere, fordi den er samlet ét sted.

Hvordan den egentlige implementering til dette ser ud, afhænger af sproget man ønsker at implementere det i og jeg vil undlade at forsøge i denne artikel, men du kan evt. kigge i denne artikel, hvis du koder i C#: Parameteriseret tekstformatering i C#, hvor jeg kommer med et bud.

Kommentar af olebole d. 25. aug 2012 | 1

<ole>

Absolut et spændende koncept! Jeg har visse overvejelser/forbehold med hensyn til den aktuelle løsning, men jeg ser det som et godt grundlag for viderudvikling. Har du overvejet at GitHub'e det?

Jeg forstår ikke helt, hvad du mener med begrebet 'placeholder' i denne forbindelse - og hvordan det adskiller sig fra begrebet 'oversættelse'.

Jeg har også lidt svært ved at se, hvordan det skal virke. Hvis dette:

```
"Der findes {0} person[#0!1:er] i {1} regist[#1!1:re|=1:er]"
```

- skal fodres med to antal, kan jeg se det fungere med tal som '1' og '2' ... altså "Der findes 1 person i 2 registre". Men hvordan skal det kunne fungere med talord: "Der findes en person i to registre"? Eller hvis vi vil accentuere, at der kun findes én enkelt: "Der findes én person i to registre"?

Den engelske version, du viser ovenfor, skal vist i øvrigt se sådan ud:

```
"{0} person[#0!1:s] exist[#0=1:s] in {1} register[#1!1:s]"
```

- men det ved jeg godt, er en lille 'tankefisk' *o)

Til gengæld ser jeg frem til, at metoden kan anvendes på alle de, der har det med at sige: "Langelinje er ca. en kilomet lang" *D

/mvh
</bole>

Kommentar af olebole d. 25. aug 2012 | 2

Glem kommentaren: "*Jeg forstår ikke helt, hvad du mener med begrebet 'placeholder' i denne forbindelse -*

*og hvordan det adskiller sig fra begrebet '*oversættelse*'" >> Det siger jo sig selv *o)

Kommentar af softspot d. 25. aug 2012 | 3

Jeg har i øvrigt overvejet muligheden for også at kunne sende tekst-værdier med til Translate-funktionen, således man f.eks. kunne lave betingelser som anvendte disse. Noget á la:

```
"{0} person[#0!en:er] findes"
```

Der kan med stor sandsynlighed findes nogle mere eksotiske eksempler på brugen af dette, men blot for at lufte tanken...

Kommentar af softspot d. 25. aug 2012 | 4

Tak for din feedback ole.

Jeg kan ikke umiddelbart gennemskue konsekvenserne af at skulle arbejde med talord, da der vel findes et relativt stort antal ord til at dække alle mulige tal :-)

Dog kan man, hvis man ved det er et begrænset antal talord der skal præsenteres, anvende betinglserne til at omsætte værdien til et talord. Nogenlunde således:

```
"Der findes [#0=0:nul|=1:én|=2:to|=3:tre|>3:mange] person[#0!1:er] i
[#1=0:nul|=1:ét|=2:to|=3:tre|>3:flere] regist[#1!1:re|=1:er]"
```

uagtet at der skal syv til mange.

Jeg må indrømme, at jeg ikke har tænkt på at GitHub'e det, dels fordi jeg, well, ikke har tænkt tanken, dels fordi jeg ikke har sat mig ind i hvordan GitHub fungerer. Hvis du har det lyst, må du da gerne bære konceptet videre. Jeg fornemmer du er mere inde i, hvad der skal til, for at få sådan noget gjort officielt tilgængeligt end jeg er...

NB: Beklager, men denne besked skulle have kommet før #3, men Eksperten synes åbenbart ikke Preview skal fixes for kommentarer til guides... :-)

Kommentar af olebole d. 25. aug 2012 | 5

Undskyld, jeg trykte mig dårligt ud :D

Jeg mente faktisk 'ordinal suffixes': 1st, 2nd, 3rd, 4th, 5th ... osv, indtil 21st, 22nd, 23rd, 24th, osv, fremtil 30'erne, 40'erne, osv.

På dansk er det bare et punktum efter tallet, men på andre sprog bruger man den slags suffixes.

Man *kunne* dække et begrænset antal - f.eks. op til 10-20 stykker. Og jeg kunne forestille mig, at man både dækkede talord og suffiks. I JS kunne det på engelsk gøres med:

```
var sOrdnSuff = "st;nd;rd;th;th;th;th;th;th;th",
aOrdnSuff = sOrdnSuff.split(";");

var sNumWrds = "one;two;three;four;five;six;seven;eight;nine;ten",
```

```
aNumWrds = sNumWrds.split(";");
```

Man kunne så parse argumenterne, der bliver sendt med til funktionen. Er de tal, indsættes de som tal (og bruges til at afgøre fler- og entalsendelser):

```
FUNC("{0} is more than {1}.", 3, 1)
-> "3 is more than 1."
```

Er de derimod strenge, parses de, og det første tegn afgør, om der skal indsættes et talord eller et tal med suffiks. Det efterfølgende tal bruges til at kalde ned i de to arrays (og til at afgøre fler- og entalsendelser):

```
FUNC("The {0} brother is older than the {1}.", "o1", "o2")
-> "The 1st brother is older than the 2nd."

FUNC("{0} is more than {1}.", "w3", "w1")
-> "three is more than one."
```

Jeg kunne ikke lige finde på et eksempel med endelser, men du forstår sikkert meningen =)

Kommentar af softspot d. 25. aug 2012 | 6

Umiddelbart ville jeg foretrække at de argumenter der medsendes ikke skal modificeres, så jeg ville da foreslå at beslutningen om suffix ligger i selve skabelonen (som det kendes fra .NET's String.Format, se bla. http://www.csharp-examples.net/string-format-int/), f.eks.

```
FUNC("The {0:o} brother is older than the {1:o}", 1, 2)
```

Alternativt skulle man indføre nogle mere komplekse udtryksformer til det eksisterende format, f.eks. en modificering af værdien der sammenlignes med inden betingelserne evalueres:

[#0%10=1:st|=2:nd|=3:rd|th]

Modificeringen (her med % for modulus) af den værdi jeg tjekker op imod sker inden de efterfølgende betingelser udføres.

Desuden indføres en fallback-værdi der gælder for alle andre tilfælde end de forgående betingelser (evaluering fra venstre mod højre og første match stopper evalueringen).

Jeg er klar over at alternativet ikke nødvendigvis gør formatet nemmere at sætte op (i modsætning til dit forslag), men det udvider i det mindste anvendelsesmulighederne lidt... :-)

Der skal nok lige tænkes lidt over, hvordan man adskiller den første modificering med værdierne, hvis modificeringen skal være meget mere kompleks end det viste :-)

Kommentar af olebole d. 25. aug 2012 | 7

Jeg tænkte bare højt - og for hurtigt. Naturligvis skal det derind og stå. 'o' og 'w' var bare eksempler *o)

Jeg foretrækker dog stadig at have de forskellige suffiks og/eller talord gemt et centralt sted - fremfor at skulle definere dem i hver oversættelse =)

Der koges videre ... =)

Kommentar af softspot d. 25. aug 2012 | 8

Enig i at standardsuffiks (eller standardformater i det hele taget) bør ligger centralt.

Mit udgangspunkt har været nogle simple regler, som kunne klare de fleste af mine behov for oversættelse og formatering af tekst og som gav oversætteren mulighed for at lave sine egne simple regler, hvor det måtte være nødvendigt.

Jeg har en kørende implementering i C#, som jeg benytter i et projekt jeg udvikler på i øjeblikket. Det er altså moderat testet på dansk og engelsk, hvor jeg indtil videre ikke er stødt i umulige scenarier. Det skal dog siges, at jeg ikke har oversat SÅ mange tekster endnu (der er vel 100-150 tekster pt.) og at disse tekster jo typisk er kortere ledetekster ifm. knapper, oversigter og labels på formularer. Jeg har dog en fornemmelse af, at det rækker til de fleste behov ifm. det aktuelle projekts behov (men jeg har jo heller ikke fået det oversat til nogle af de lidt mere eksotiske sprog endnu :-)).

Kommentar af jensenjs d. 27. apr 2013 | 9

Jeg lavede en gang for maaaaaaange år siden 1978-1980 et lignende projekt i BASIC på min ABC80.
Jeg fik faktisk løst langt de fleste problemer :-)
Jeg ville gerne liste den op her men den ligger på kasettebånd sammen med en hel del programmer, jeg har en ABC80 emulator men kan desværre ikke overfører båndene til mine PC'er

Jeg er en habil programmør på både BACIS og C++
Men ikke så god.

Kommentar af jensenjs d. 27. apr 2013 | 10

Glemte lige, godt skrevet :-)

Kommentar af softspot d. 29. apr 2013 | 11

Tak for roserne :-)

Jeg har faktisk et par løsningsforslag liggende (JavaScript og C#), men jeg mangler bare lidt tid til, at få dem publiceret her på Eksperten. Jeg håber de kan komme på snart :-)

Kommentar af jokkejensen d. 02. maj 2013 | 12

syntes du bør holde javascript ude af denne, søgemaskinerne afvikler ikke dette. Det bør ske serverside imo.

Thumbs up herfra.

Kommentar af softspot d. 03. maj 2013 | 13

Tak Jokke! :-)

JavaScript finder også anvendelse i andre kontekster end dem, hvor der skal tænkes på SEO (f.eks. SPA og ved brug af AJAX i øvrigt) og i disse sammenhænge kunne det være rart nok, at have mulighed for at formatere sine tekster med JS på klienten.

Kommentar af arne_v d. 19. maj 2013 | 14

Jeg har foreslaaet en alternativ syntax som kommentar til C# artiklen.