

i18n en PHP

Carlos Sánchez - carlos@nvivo.es
David Gaya - david.gaya@assertio.es



Introducción

- **i18n** NO ES sólo presentar una web en varios idiomas
- **i18n** is the process of designing a software application so that it can be adapted to various languages and regions without engineering changes (Wikipedia)
- **L10n** - Enfoca aspectos culturales (Fechas, monedas, medidas...)
- **L10n** is the process of adapting software for a specific region or language by adding **locale**-specific components and translating text. (Wikipedia)



Caso il8n de nvivo.es

- Nginx, PHP4 + Smarty, MySQL 5, JQuery
Plataforma completa en ISO-8859-1
- Despliegue de **nvivo.es** en más países.
Comenzando con UK y DE
- **Información Local**
- Localización geográfica: Webservice de **Geonames**. Múltiples dominios, TLDs diferentes por cada país, URLs diferentes en cada idioma:
<http://www.5gig.co.uk>
<http://www.5gig.de>



Pasos realizados

- Paso de todos los archivos de iso-8859-1 a **UTF-8**. `iconv --from-code=ISO-8859-1 --to-code=UTF-8 $file > $nfile`
- Configuración de Nginx, php.ini, MySQL.
- Solución de problemas UTF-8 de PHP: Funciones **mb_string**
- Fechas, moneda, unidades, soporte de localidades internacional: **Geonames, strftime**
- Instalación de **locales** en el sistema, **gettext**, utilidades gettext.

Pasos realizados II

- Creación de estructuras de catálogos de idioma: archivos **.PO**, **.MO**
- Uso de gettext en PHP, uso de gettext en Smarty.
- Traducción de los catálogos.
- Traducción de las URLs y gestión con múltiples **.htaccess** (rewrite de nginx).



UTF-8

- ¿qué es UTF-8? Es una norma de transmisión de longitud variable para caracteres codificados utilizando Unicode
- **Unicode** es un estándar industrial cuyo objetivo es que cualquier texto en cualquier forma e idioma pueda ser codificado para el uso informático
- **Norma nº 1 - TODO EN UTF-8:** Editor, archivos, BD, tablas, campos...
- Soluciona el **90%** de tus problemas de i18n
- ~~ISO-8859-1~~



A tener en cuenta: Backend

- locale

```
locale -a //vemos todos los locales
/etc/locale.gen //Añadimos el locale que queremos instalar
locale-gen //genera los locales que encuentra en /etc/locale.gen
```

- gettext

```
gettext("Hello World");
sprintf(gettext("Hello %s"),$var1);
_("This way better");
```

- MySQL

```
ALTER DATABASE Journal SET CHARACTER SET utf8;
ALTER TABLE Journal.Posts DEFAULT CHARSET SET utf8;
ALTER TABLE MODIFY body body TEXT CHARACTER SET utf8;
CHARACTER SET utf8 COLLATE utf8_general_ci;

SET NAMES 'utf8';
```

- Apache

```
.htaccess AddDefaultCharset UTF-8
```



A tener en cuenta: PHP

- PHP4, PHP5 y el UTF-8 no son muy amigos
¡funciones **mb_string**! ¿ **PHP 6** ?
- ~~utf8_encode, utf8_decode~~
No es necesario si se usa UTF-8, excepto con **strftime** :(
- <http://www.phpwact.org/php/i18n/utf-8>

A tener en cuenta: Frontend

- XHTML

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

- Smarty Gettext

```
<p>{t}Hello World{/t}</p>  
<p>{t 1=$var1 escape="no"}Hello <strong>%1!</strong>{/t}</p>
```

- Javascript

Jquery gettext ¿? - Problemas de rendimiento (AJAX, JSON)

Nosotros preferimos Smarty

Ojo con las **expresiones regulares** y el paso a UTF-8 de ficheros .js



gettext

- Flexible: Fácil de actualizar nuevas frases
- GPL, Estándar de facto, herramientas, utilidades
- ¿por qué reinventar la rueda?
`xgettext, msgfmt, msgmerge....`
- La opción menos mala
`xgettext -c archivo.php`
`msgfmt messages.po`
`msgmerge old.po messages.po -o new.po`
- POEdit y KBabel



iA programar!

Host - 217.113.247.140 / phpbarcelona.phpbcn.org

User - phpworkshop

Passwd - phpworkshop

