Handling special characters in links

Handling special characters in links

Not all shops use plain ASCII text content. Any shop which uses product names with special characters could find itself needs to provide proper support for those special characters in link.

In effect, even the "basic" languages now integrate loan words from other languages: any English speaker knows the difference between "resume" and "résumé".

Therefore, you should always strive to support special characters.

Many special characters are not handled natively by PrestaShop, mostly those from languages that use a non-Latin alphabet. But since PrestaShop is built around UTF-8 (a Unicode encoding), you can easily bring support to the default behavior.

In order to support your language's special characters within your links using URL rewriting, you must override some methods.

In this page, you will see how to handle the Cambodian language (Khmer).

The presented method uses the PCRE library (http://www.pcre.org/pcre.txt), which is included in PHP. It is a regular expression library which can have many uses, among which is helping with support for special characters. It can help you support many other languages, or "scripts".
The current list is of scripts is: Arabic, Armenian, Avestan, Balinese, Bamum, Batak, Bengali, Bopomofo, Brahmi, Braille, Buginese, Buhid, Canadian_Aboriginal, Carian, Chakma, Cham, Cherokee, Common, Coptic, Cuneiform, Cypriot, Cyrillic, Deseret, Devanagari, Egyptian_Hieroglyphs, Ethiopic, Georgian, Glagolitic, Gothic, Greek, Gujarati, Gurmukhi, Han, Hangul, Hanunoo, Hebrew, Hiragana, Imperial_Aramaic, Inherited, Inscriptional_Pahlavi, Inscriptional_Parthian, Javanese, Kaithi, Kannada, Katakana, Kayah_Li, Kharoshthi, Khmer, Lao, Latin, Lepcha, Limbu, Linear_B, Lisu, Lycian, Lydian, Malayalam, Mandaic, Meetei_Mayek, Meroitic_Cursive, Meroitic_Hieroglyphs, Miao, Mongolian, Myanmar, New_Tai_Lue, Nko, Ogham, Old_Italic, Old_Persian, Old_South_Arabian, Old_Turkic, Ol_Chiki, Oriya, Osmanya, Phags_Pa, Phoenician, Rejang, Runic, Samaritan, Saurashtra, Sharada, Shavian, Sinhala, Sora_Sompeng, Sundanese, Syloti_Nagri, Syriac, Tagalog, Tagbanwa, Tai_Le, Tai_Tham, Tai_Viet, Takri, Tamil, Telugu, Thaana, Thai, Tibetan, Tifinagh, Ugaritic, Vai, Yi.

Most of these languages are handled with the `\pL` PCRE shortcut. It is therefore recommended to test your PrestaShop installation to see if your language is not already handled by the default behavior. If not, you can test with the list above.

Overriding

In order to handle Cambodian chars, you will need to add a piece of PCRE code to allow these chars to match with regular expressions in PrestaShop.

That piece of PCRE code is quite simple: `\p{Khmer}`

The first class which we have to override is in the `Validate` class. We're looking to work with `isLinkRewrite()` method. This method will validate that the string is a valid URL (without code injection).

```
public static function isLinkRewrite($link)
{
    if (Configuration::get('PS_ALLOW_ACCENTED_CHARS_URL'))
        return preg_match('/^[_a-zA-Z0-9\-\pL\p{Khmer}]+$/u', $link);
    return preg_match('/^[_a-zA-Z0-9\-]+$/', $link);
}
```

Then, you have to override the `Tools` class, and more precisely the `str2url()` method, which allows you to clean a string and turn it into a valid and safe URL.

```
public static function str2url($str)
{
    if (function_exists('mb_strtolower'))
        $str = mb_strtolower($str, 'utf-8');

    if (!function_exists('mb_strtolower') || !Configuration::get('PS_ALLOW_ACCENTED_CHARS_URL'))
        $str = Tools::replaceAccentedChars($str);

    // Remove all non-whitelist chars.
    if (Configuration::get('PS_ALLOW_ACCENTED_CHARS_URL'))
        $str = preg_replace('/[^a-zA-Z0-9\s\'\:\/\[\]-\pL\p{Kmer}]/u', '', $str);
    else
        $str = preg_replace('/[^a-zA-Z0-9\s\'\:\/\[\]-]/','', $str);

    $str = preg_replace('/[\s\'\:\/\[\]-]+/', ' ', $str);
    $str = str_replace(array(' ', '/'), '-', $str);

    // If it was not possible to lowercase the string with mb_strtolower, we do it after the transformations.
    // This way we lose fewer special chars.
    if (!function_exists('mb_strtolower'))
        $str = strtolower($str);

    return $str;
}
```

And finally, you will need to update the default routes from the `Dispatcher` class. This is done by adding the following code at the end of the `Dispatcher` class, after the "`$this->loadRoutes();`" line:

```
public function __construct()
{
    // ...
    // previous code does not change
    // ...

    $this->loadRoutes();

    foreach ($this->default_routes as &$routes)
        foreach ($routes['keywords'] as &$keywords)
            $keywords['regexp'] = str_replace('\\pL', '\\pL\\p{Khmer}', $keywords['regexp']);
    parent::__construct();
}
```

You have now overridden the main methods, and your PrestaShop installation is now able to handle Cambodian chars in your URL.

If you want to handle another language, you just need to add a new condition to your regexp.

Sample use

Using another language than Khmer

Let's say that instead of Khmer, we want to support the Telugu language, which is mostly used in south-central India.

All we need to do is change the code above, and replace every instance of \p{Khmer} with \p{Telugu}:

For instance, the first override would be:

```
public static function isLinkRewrite($link)
{
    if (Configuration::get('PS_ALLOW_ACCENTED_CHARS_URL'))
        return preg_match('/^[_a-zA-Z0-9\-\pL\p{Telugu}]+$/u', $link);
    return preg_match('/^[_a-zA-Z0-9\-]+$/', $link);
}
```

Adding support for another language

Let's say that in addition to Khmer, we want to support Glagolitic, a Slavic alphabet from the 9th century.

In this case, we simply have to repeat the PCRE code, with "Glagolotic" instead of "Khmer".

For instance, the first override would be:

```
public static function isLinkRewrite($link)
{
    if (Configuration::get('PS_ALLOW_ACCENTED_CHARS_URL'))
        return preg_match('/^[_a-zA-Z0-9\-\pL\p{Khmer}\p{Glagolitic}]+$/u', $link);
    return preg_match('/^[_a-zA-Z0-9\-]+$/', $link);
}
```

As you can see, we have both `\p{Khmer}` and `\p{Glagolitic}` in the regexp, one after the other. You can add as many languages as needed.