i18n made easy

illustration with CherryPy

Nicolas DERAM

nderam@itaapy.com

Introduction: Who is the speaker

Nicolas DERAM

- Python developper for 2 years.
- Software engineer at Itaapy, a (French) company specialized on Web applications development.
- Contributor on the itools library, author of the module ical implementing the iCalendar specifications.

Introduction

- *i18n* = Internationalization

  It is the operation by which a program [or a set of programs turned into a package] is made aware of and able to support multiple languages.

- Illustration with *CherryPy*

  CherryPy is a pythonic, object-oriented web development framework which does not deal with tasks such as templating. (Version: 2.2.1)

- Using tools from *itools*

  itools is a python library which provides a rich API to perform various tasks. (Version: 0.13)

From one to many languages

- Example : only one language
- Translation work
- Example : multiple languages

Example

Only one language

Example: one language (1/4)

(Firefox)

Example: one language (2/4)

```
import cherrypy

class Root:
    @cherrypy.expose
    def index(self):
        return open('./example.xml').read()

    @cherrypy.expose
    def doLogin(self, username=None, passwd=None):
        if username == 'toto' and passwd == 'toto':
            return 'Hello World !'
        return 'Authentication error!'

cherrypy.root = Root()
cherrypy.server.start()
```

Example: one language (3/4)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head></head>
 <body>
  <form action="doLogin" method="post">
   <p> Username</p>
   <input type="text" name="username" value="" />
   <p> Password</p>
   <input type="password" name="passwd" value=""/>
   <p><input type="submit" value=" Login"/></p>
  </form>
 </body>
</html>
```

Example: one language (4/4)

Files obtained:

```
.
|-- example.py
‘-- example.xml
```

Example

Translating work

Translating work

Two kinds of messages to translate:

- From the code:
  - string ’Hello World !’
  - string ’Authentication error !’

- From the template:
  - content values ’Username’ and ’Password’
  - attribute value ’Login’

Example

multiple languages

Example

multiple languages

- Extracting messages
- Translating messages
- Building files
- Connecting translated files to the application
- Selecting language
- Runtime

Extracting messages (1/4)

Extracting messages from the code

- Use Unicode for all string to be translated

For our example:

```
@cherrypy.expose
def doLogin(self, username=None, passwd=None):
    if username == 'toto' and passwd == 'toto':
        return u'Hello World !'
    return u'Authentication error!'
```

Extracting messages (2/4)

Extracting messages from the template

- Nothing to do !!
  - No special attribute to specify in the template
  - No code to add in the template

Extracting messages (3/4)

Generate .pot files to localize the application.

```
$ igettext-extract example.py example.xml.en \
    > locale/locale.pot
$ cat locale.pot
...
#: example.py:13
msgid "Authentication␣error␣!"
msgstr ""
#: example.py:12
msgid "Hello␣World␣!"
msgstr ""
#: example.xml.en:0
msgid "Login"
msgstr ""
...
```

Extracting messages (4/4)

For practical reason,

- we use a directory "locale",
- we rename xml file into xml.en as it is our reference language.

```
.
|-- example.py
|-- example.xml.en
‘-- locale
    ‘-- locale.pot
```

Example

multiple languages

- Extracting messages
- Translating messages
- Building files
- Connecting translated files to the application
- Selecting language
- Runtime

Translating messages (1/2)

Generate .po file for each language.

```
$ igettext-merge locale/locale.pot locale/fr.po \
    > locale/fr.po
```

Translate messages into .po files :

```
#: example.py:13
msgid "Authentication error !"
msgstr "Erreur d'authentification !"
#: example.py:12
msgid "Hello World !"
msgstr "Salut Monde !"
#: example.xml.en:0
msgid "Login"
msgstr "Valider"
...
```

Translating messages (2/2)

We get :

```
.
|-- example.py
|-- example.xml.en
‘-- locale
    |-- fr.po
    ‘-- locale.pot
```

Example

multiple languages

- Extracting messages
- Translating messages
- Building files
- Connecting translated files to the application
- Selecting language
- Runtime

Building files (1/3)

Code side:

Generate .mo file for each language to be able to get translation at runtime.

```
$ msgfmt locale/fr.po -o locale/fr.mo
```

Building files (2/3)

Templates side:

Generate xml file for each language.

```
$ igettext-build example.xml.en locale/fr.po
    > example.xml.fr
  ...
  <body>
    <form method="post" action="doLogin">
      <p> Identifiant</p>
      <input name="username" type="text" value=""/>
      <p> Mot de passe</p>
      <input name="passwd" type="password" value=""/>
      <p><input type="submit" value=" Valider"/></p>
    </form>
  </body>
```

Building files (3/3)

Now we have :

```
.
|-- example.py
|-- example.xml.en
|-- example.xml.fr
‘-- locale
    |-- fr.mo
    |-- fr.po
    ‘-- locale.pot
```

Example

multiple languages

- Extracting messages
- Translating messages
- Connecting translated files to the application
- Selecting language
- Runtime

Connecting translated files (1/2)

The files we built, containing all translated files, constitute a domain.

itools offers an API to handle domains.

So we just connect our application to our domain "locale" by specifying its path.

Connecting translated files (2/2)

```
import cherrypy
from itools.gettext.domains import DomainAware
from itools.gettext.domains import register_domain

class Root( DomainAware):
    class_domain = 'Example'

    def index(self):
        ...
    def doLogin(self, username=None, passwd=None):
        ...

 register_domain(Root.class_domain, './locale')
     ...
```

Example

multiple languages

- Extracting messages
- Translating messages
- Connecting translated files to the application
- Selecting language
- Runtime

Set available languages

Now we are linked to our domain, we have to know which data we want to get.

First we need to specify the available languages for our application.

```
...
class Root(DomainAware):
    class_domain = 'Example'
    @classmethod
    def get_languages(cls):
        return ['en', 'fr']
  ...
```

Select the language

We need to select the language to be used. To do this, we add a method which get information from the client browser.

```
from itools.i18n.accept import AcceptLanguage
class Root(DomainAware):
  ...
  @classmethod
  def  select_language(cls, languages=None):
     if languages is None:
         languages = cls.get_languages()
     accept = cherrypy.request.headers['Accept-Language']
     accept = AcceptLanguage(accept)

     return accept.select_language(languages)
  ...
```

Example

multiple languages

- Extracting messages
- Translating messages
- Connecting translated files to the application
- Selecting language
- Runtime

Runtime (1/2)

Code side:

```
import cherrypy
from itools.gettext.domains import DomainAware

class Root( DomainAware):
    ...
    @cherrypy.expose
    def doLogin(self, username=None, passwd=None):
        if username == 'toto' and passwd == 'toto':
            return self.gettext(u'Hello World !' )
        return self.gettext(u'Authentication error!' )
    ...
```

Runtime (2/2)

Template side:

We call "select_language()" method while specifying the template to be used.

```
...
class Root(DomainAware):
    ...
    @cherrypy.expose
    def index(self):
        language = self.select_language()
        return open('./example.xml.%s' % language).read()
  ...
```

Multilingual Example

(Firefox)

Conclusion

Using itools API and scripts has some advantages to make a multilingual application:

- Reduces the development costs. Also, experience has proven that adding explicit mark-up is a tedious and very error-prone task.
- Reduces the performance overhead because it removes the need to do a lookup call for every sentence.
- `itools.i18n` and `itools.gettext` can be used with any framework.

References

- CherryPy Web Site, http://www.cherrypy.org
- itools Web Site, http://www.ikaaro.org/itools
- itools mailing list, http://mail.ikaaro.org/mailman/listinfo/itools
- Contact : Nicolas Deram, nderam@itaapy.com