Internationalizing your Django project

Matt Croydon

PyCon 2010

Necessary definitions

What is internationalization?

"The goal of internationalization is to allow a single Web application to offer its content and functionality in multiple languages and locales."

—Django Internationalization documentation

What is localization?

"Localization is the process of adapting internationalized software for a specific region or language by adding locale-specific components and translating text."

http://en.wikipedia.org/wiki/Internationalization_and_localization

What's a locale?

"...a set of parameters that defines the user's language, country and any special variant preferences that the user wants to see in their user interface."

http://en.wikipedia.org/wiki/Locale

Common abbreviations

- Internationalization: i18n
- Localization: l10n

Why should I internationalize?

- You likely have users outside of your country with localization needs.
- Particularly important in open source and reusable applications.
- It's much easier to internationalize a small codebase than a large one.

Marking strings for translation

```
from django.utils.translation import ugettext

# Before i18n
output = "Hello World!"

# After i18n
output = ugettext("Hello World!")
```

Shortcut

```
from django.utils.translation import ugettext as _

output = _("Hello World!")
```

Translating views

```
from django.http import HttpResponse
from django.utils.translation import ugettext as _

def my_view(request):
    output = _("#daniellindsleyrocksdahouse")
    return HttpResponse(output)
```

It's just Python

- gettext, ugettext, ungettext, etc. are just simple wrappers around built-in Python behavior.
- Django provides a lazy translation layer on top, useful in models and template tags/ filters.

Translating models

```
from django.db import models
from django.utils.translation import \
ugettext_lazy as _

class MyThing(models.Model):
    name = models.CharField(_('name'),
        help_text=_('This is the help text'),
        max_length=200)
    class Meta:
        verbose_name = _('my thing')
        verbose_name_plural = _('my things')
```

Pluralization

```
from django.http import HttpResponse
from django.utils.translation import ungettext

def hello_world(request, count):
    page = ungettext('there is %(count)d object',
        'there are %(count)d objects', count) % {
        'count': count,
    }
    return HttpResponse(page)
```

Translating templates

```
{% load i18n %}

<title>{% trans "This is the title." %}</
title>
<title>{% trans myvar %}</title>
```

Blocktrans

```
{% load i18n %}

{% blocktrans %}
This string will have {{ value }} inside.
{% endblocktrans %}
```

Blocktrans with filters

```
{% load i18n %}

{% blocktrans with book|title as book_t and
author|title as author_t %}
This is {{ book_t }} by {{ author_t }}
{% endblocktrans %}
```

Blocktrans pluralization

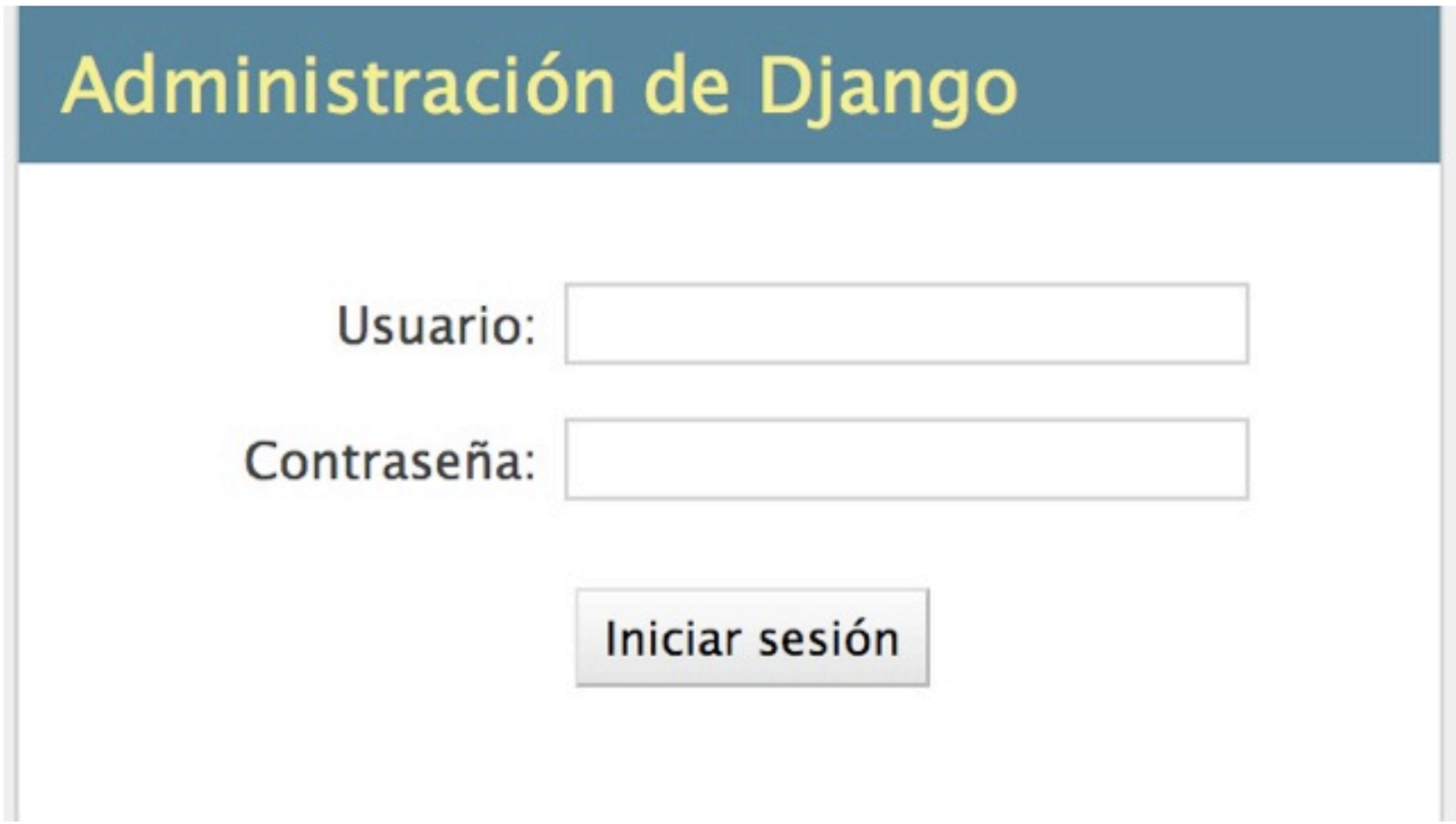
```
{% load i18n %}

{% blocktrans count list|length as counter %}
There is only one {{ name }} object.
{% plural %}
There are {{ counter }} {{ name }} objects.
{% endblocktrans %}
```

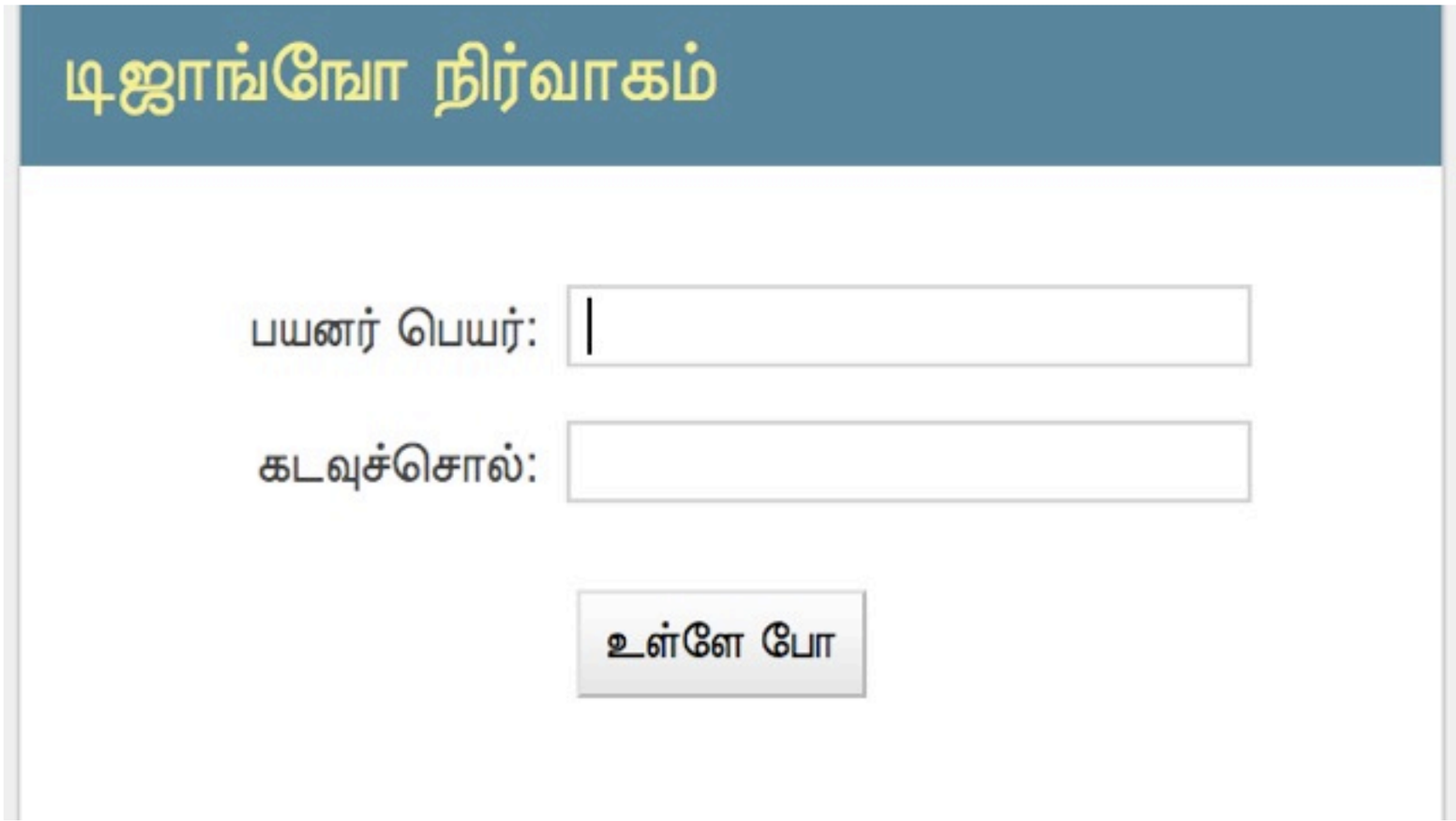
Enabling i18n

```
# myproject/settings.py

MIDDLEWARE_CLASSES=(
 'django.middleware.common.CommonMiddleware',
 'django.contrib.sessions.middleware.SessionMiddleware',
 'django.middleware.locale.LocaleMiddleware',
 'django.contrib.auth.middleware.AuthenticationMiddleware',
)
```



Admin in Spanish



Admin in Tamil

Which language to serve?

1. `django_language` session key
2. `django_language` cookie
3. `Accept-Language` HTTP header
4. `LANGUAGE_CODE` setting

Creating message files

- `makemessages` management command.
- Run it from the root of your project.
- Or run it from the root of your app.
- Django looks for translations in both of those locations.
- Make sure you have gettext installed.

Creating message files

```
$ cd /path/to/myproject
$ mkdir locale
$ django-admin.py makemessages -l de
$ django-admin.py makemessages -l es -e
html,txt
```

Message files (.po)

```
#: things/models.py:5
msgid "name"
msgstr ""

#: things/models.py:6
msgid "This is the help text"
msgstr ""

#: things/models.py:8
msgid "my thing"
msgstr ""
```

Doing the translation

```
#: things/models.py:5
msgid "name"
msgstr "nombre"
```

Compiling message files (.mo)

```
$ django-admin.py compilemessages
```

Before translation

After translation

Setting the language manually

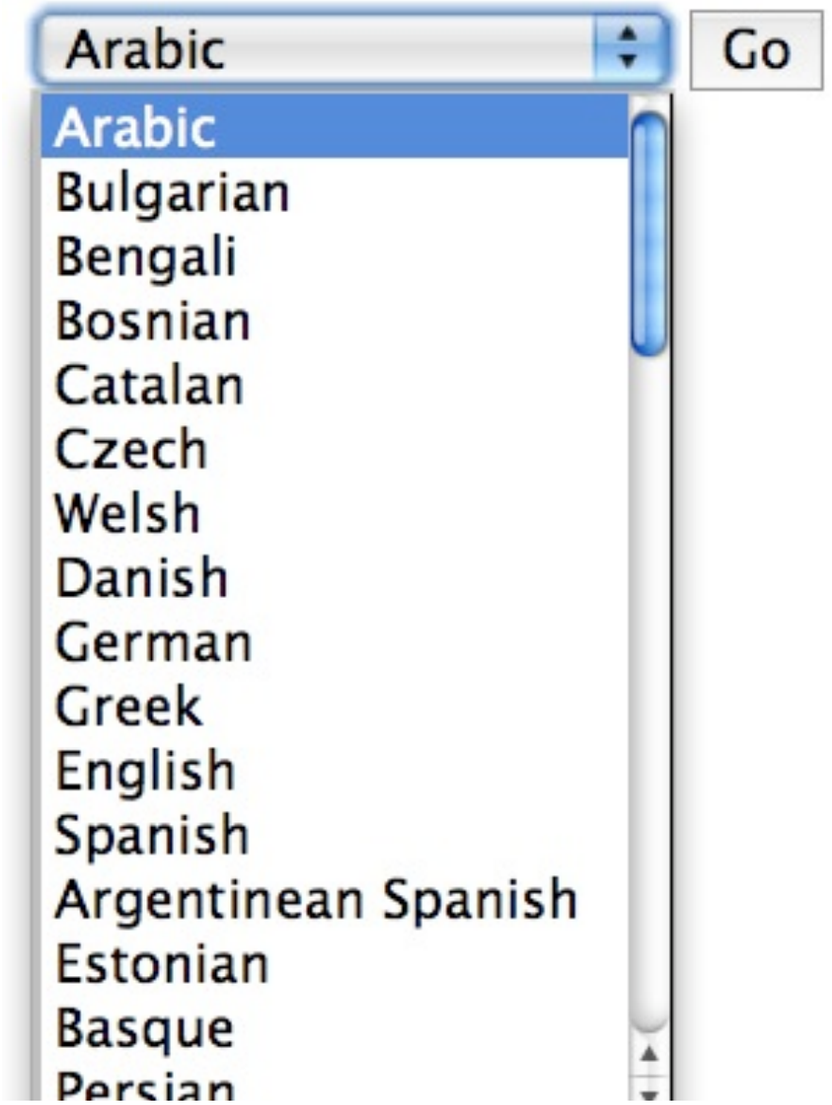
```
from django.conf.urls.defaults import *
from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    (r'^admin/(.*)', admin.site.root),
    (r'^i18n/', include('django.conf.urls.i18n')),
    (r'^set_language/$',
        'django.views.generic.simple.direct_to_template',
        {'template' : 'set_language.html'}),
)
```

Example form

```
<form action="/i18n/setlang/" method="post">
<input name="next" type="hidden" value="/next/page/" />
<select name="language">
{% for lang in LANGUAGES %}
<option value="{{ lang.0 }}">{{ lang.1 }}</option>
{% endfor %}
</select>
<input type="submit" value="Go" />
</form>
```

Example form in action



Advanced localization coming in Django 1.2

- Set `USE_L10N = True` in settings.
- Enables locale-aware formatting of dates and times

Third-party apps

- django-rosetta for translation in the admin
- django-transmeta, django-multilingual, django-pluggable-model-i18n, django-modeltranslation, transdb for translating dynamic content
- Many more

Learning More

Extensive internationalization documentation at docs.djangoproject.com

Questions?