e-cidadania

e-democracy tool for citizens

Documentation

Release

Cidadania S. Coop. Galega

April 05, 2013

Contents

e-cidadania is an open-source e-democracy web tool intended for citizen participation.

e-cidadania was designed with complex participative processes in mind, lile a participative budget where people can decide what to do with a part of the annual bugets of a region or city, where making a simple meeting and disordered proposals doesn't meet the expectations.

This software is based on the django framework and some external libraries.

> **Warning:** e-cidadania is in heavy development, because of that some parts of it may suffer changes, especially the data models and documentation until some releases.

Contents 1

Installation & Configuration

1.1 Installation

e-cidadania installation is very simple and is done almost in the same as any other django platforms.

1.1.1 Downloading platform

Official download page

Note: The download section in the official website is not available yet.

The are several ways to download e-cidadania. The most simple of them is going to the downloads page in the website and download the latest stable or development versions, ready to use.

GitHub packages

Other way of downloading it is through the Github downloads page, which autogenerates a .zip and .tar.gz files based on the repository tags. You can find it in:

```
https://github.com/cidadania/e-cidadania/tags
```

From repository

See *Development version*

Stable version

You can find the latest stable version in the download page in ecidadania.org:

```
http://ecidadania.org/en/downloads
```

Development version

Development version is available through various places. We use GIT as version control system, so you will have to install it in your computer.

GitHub *(official repository)*:

```
git clone git://github.com/cidadania/e-cidadania.git
```

Gitorious: *(secondary repository)*:

```
git clone git://gitorious.org/e-cidadania/mainline.git
```

Repo.or.cz *(official mirror)*:

```
git clone git://repo.or.cz/e_cidadania.git
```

1.1.2 Installing

> **Warning:** Since e-cidadania 0.1.5 we included an automated buildout system. If you are going to develop in e-cidadania you should take a look to the *Creating the environment*.

The installation process for e-cidadania is quite simple.

Requirements

- Apache, nginx, or any other web server with CGI suppport
- FastCGI, uWSGI, Passenger or other CGI.

Dependencies

- Python 2.7.x
- django 1.4.x
- PIL *(Python Imaging Library)*
- python-datetime *(version 1.5)*
- django-tagging
- django-grappelli
- feedparser
- pyyaml

You can install all the required dependencies automatically with this command:

```
# pip install -r requirements.txt
```

Most of the requirements are automatically installed this way, but there are some packages that need to be installed via the system packages, for example:

- PIL

> **Warning:** There are reported errors for people that tried to install PIL from pip instead from the official system package. The problem is due to the lack of some features of PIL needed in e-cidadania.

Platform

There isn't a proper installation process in e-cidadania, you just have to copy the files to you preferred installation directory.

If you are going to use it in production, or you just want to give e-cidadania a try follow this steps:

Note: e-cidadania comes preconfigured for a development environment. You will have to set the DEBUG flag to **False** in *src/e_cidadania/settings/__init__.py*

```
$ ./manage.py syncdb # This will create all the database objects
$ ./manage.py collectstatic # This will copy all the static content to *static/*
$ ./manage.py runserver
```

This last command will execute the development server in the port 8000 of your machine, so you just need to type ******localhost:8000" inside a web browser.

Now you can continue to *Configuration*

1.2 Configuration

The e-cidadania platform is almost ready-to-use after unpacking, but you will have to edit the *settings.py* file if you want to install it in production. We explain here all the settings realted with the initial configuration of e-cidadania. If you want more detailed information about the settings, please take a look at the official django documentation at **'http://docs.djangoproject.com'_**

The e-cidadania setting have been splitted into four major files:

```
settings/
    __init__.py
    defaults.py
    development.py
    production.py
```

1.2.1 __init__

This file loads one configuration or another based on the *DEBUG* flag. You have to set this first.

DEBUG (boolean) You have to set this value to *True* or *False* depending if you are using the platform or want to debug the e-cidadania platform.

1.2.2 defaults

This file stablishes the common settings across the both environments. Most of this settings are e-cidadania specific, we will specify here the ones you can modify safely.

SECRET_KEY (hash) It's **obligatory** to modify this value before deployment. This key is used for generating the CSRF and some other security features in django.

MEDIA_ROOT (directory) e-cidadania comes with this directory set by default for development but you will have to modify it to suit the directory where you media content or user uploaded content will be.

MEDIA_URL (URL slug) This will be the accesible URL for the media content. e-cidadania comes set to "uploads/" but you can modify it at any time.

STATIC_ROOT (directory) This directory works the same as MEDIA_ROOT but for static content (JavaScript, CSS, images, etc.). e-cidadania comes with it set by default for development but you will have to modify it to where you static content will be stored.

STATIC_URL (URL slug) This will be the accesible URL for the static content. e-cidadania comes with it set to "static/" but you can modify it at any time.

ACCOUNT_ACTIVATION_DAYS (integer) This is the number of days that you will allow to your users for verifying their accounts after they registered into the platform. *Default: 2*

FIRST_WEEK_DAY (integer) Just in case your country calendars don't start in Monday, you have the possibility of making the week start on Sunday (put 1 instead of 0). *Default: 0 (Monday)*

This are the main settings that you will have to modify to make the deployment of e-cidadania, you shouldn't need to modify the rest unless you want a very specific deployment, in which case we recommend you to read the django documentation about the settings.

1.2.3 development and production

development.py and *production.py* are minor configuration files with all the parameters we think you will need to make a development or production server.

Database

Configuring the database:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': 'db/sqlite.db',
        'USER': '',
        'PASSWORD': '',
        'HOST': '',
        'PORT': '',
    }
}
```

First of all will be to set up the database. By default e-cidadania is set up to use a local SQLite3 database, which will be useful to make tests, but we don't recommend to use it in production.

An example of a database on a DreamHost shared server is this:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'superb_database',
        'USER': 'databaseadmin',
        'PASSWORD': 'somepassword',
        'HOST': 'mysql.myserver.org',
        'PORT': '',
    }
}
```

DEBUG (Boolean True, False) The debug mode is meant for development only, it overrides some of the security systems of django and offers a detailed output when django crashes. This setting is meant to not be modified by the user.

TIMEZONE Time zone that e-cidadania will use as default.

LANGUAGE_CODE Main language that e-cidadania will use for the users. It uses the language code format, p.e. en-US (United States english)

CACHES Configure the cache backcend for django. Please refer to the django documentation so you can select a cache according to your needs.

ADMINS (python dictionary) List of *name*, *email* tuples with the administrators data. This is used in case e-cidadania has to send some report or django sends an error log.

EMAIL SETTINGS The email settings are pretty straightforward, so we will not explain them here.

> **Warning:** Django 1.4 still doesn't have nice support for autoverified SSL emails, you will need to use secure (but not handshaked) TLS email addresses.

1.2.4 Extensions

Note: This section is still on development.

Extensions are django applications that you can attach to e-cidadania to improve its functionalities. Currently e-cidadania doesn't support a hotplug plugin system.

You can continue now to *Deployment*

1.3 Deployment

1.3.1 Installing extra modules

e-cidadania comes with a basic set of modules that allow to make a complete participative process, but you may want to install other modules to have new features, like forums, chats, or any other kind of django application.

Django does not support module hotplugging, nor e-cidadania. You will have to install all the modules you think you will need before putting e-cidadania in production.

1.3.2 Apache 2

Note: This section is still on development.

1.3.3 nginx + uWSGI

Note: The installation through uwsgi works for other servers, not just nginx, since it's uWSGI who executes the e-cidadania instance while the server just serves the static content.
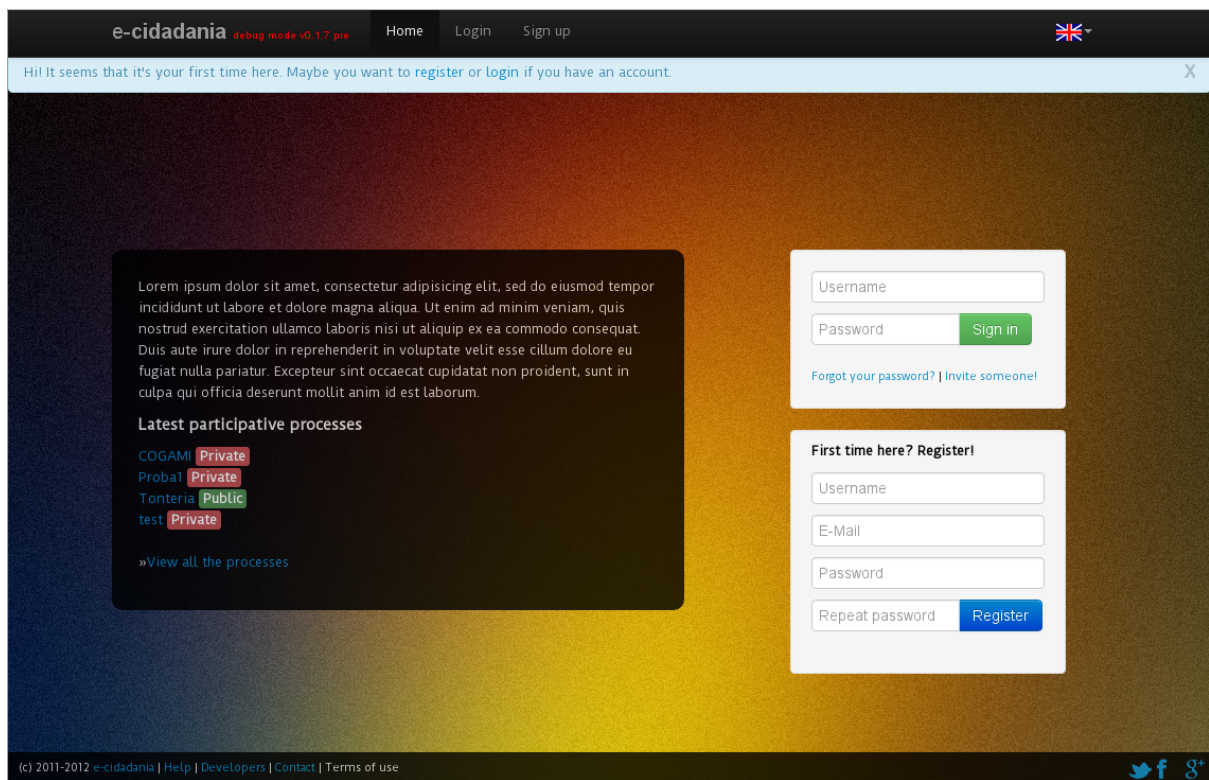
Note: This section is still on development.

1.3.4 DreamHost

The instructions for deployment on a DreamHost shared server are too extensive to include here. You can find them in the Oscar Carballal blog

Manuals

2.1 User manual

2.1.1 What is e-cidadania

e-cidadania is an e-democracy platform for citizen participation. Its main goal was to support participative budgets, but as we approached to it, we were able to do it flexible enough to be used in other citizen participation aspects as well.

2.1.2 How to register

For registering in e-cidadania you just have to fill the form in the index page and click "Register". An email will be sent to you for activating the account. Once you do it you will be an e-cidadania user, able to participate in the

public and private processes and even create your own ones.

After that you just have to put your username and password in the upper form in the index page and click "Log in".

2.1.3 Participating in a process

How to participate in a process

There are two types of processes, *public* and *private*. For participating in a public process you just have to be registered in the platform, although you can view all the activities in the process without being registered, but you won't be able to participate in it.

For participating in private processes you will be presented a page with a button called "I want to participate". You just have to click on it and an email will be sent to the space administrators so they can approve or not your participation in the process.

Processes

What are they?

Processes are places where the citizen participation takes place. There can be debates, votings, news, events, documentation, etc.

Internally, they are designated as "spaces" where people can participate. Being a registered user in the platform allows you to create as many processes as you like and be their "space administrator".

How do they work

A process can contain several modules or just one of them. A module is usually a functionality inside e-cidadania, for example, the voting module allows users to vote proposals or other kinds of stuff.

If you are a space administrator and you want to know how to use your administrator rights, please visit the *Administration manual*

Proposals

How to send a proposal

Sending proposals is very easy. You just have to access to your participative space and click on "Proposals", after that you need to click on "Add new proposal" on the right column.

A simple form will appear with several fields:

Title The proposal title. It must be a sinthesis of the proposal.

Description An extensive description of the proposal. You can link various external elements like images, websites, videos, etc.

Latitude/Longitude Right now e-cidadania has a simple geolocation sustem. If you want your proposal to be geolocated to need to put the coordinates.

Tags Tags make proposal identification easier, for example:

```
Title: The Apple Store sidewalk is broken
Tags: broken, sidewalk, apple, store
```

Debates

How to join a debate

The debates system is new, so pay attention to not get lost, this is not like an internet forum.

We have moved in-person debate models to this platform, and one of them is the one we're going to show you. You'll see how fast you get used!

Voting

News

Comments

File repository

To see or download any document you just have to click on the document and it will start downloading. If the document tou want doesn't appear in the list you just have to make click on "See all documents" and a page will come up with all the documents related to that participative process.

2.1.4 Frequently Asked Questions

If you have any frequently asked questions that doens't appear here please tell us!

2.2 Administration manual

This is a small introductory manual to teach you about how to use e-cidadania without messing everything up :).

2.2.1 Administration panel

2.2.2 Users

Restricting user registration

Note: This behaviour will change on e-cidadania 0.2

---

e-cidadania has a basic automated registration system that the administrator can activate or deactivate at his own will. By default, e-cidadania comes with registrarion activated, but if for some reason the administrator wants to block that registration system (passing to manual mode) he just has toactivate when he considers removing the comment symbol (sharp).

apps/userprofile/urls/en.py:107:

```
# url(r'^register/$', register, name='signup'),
```

If the platform is well set up, the registration system should take care of everything.

Permissions

Permissions in e-cidadania are inherited directly from the django auth system. This way we have group based permissions and user based permissions.

Note: e-cidadania 0.2 will have a row-level permission system.

There are four basic permissions:

Creation Depending on the moderation grade that you have been given, you can add simple content or more complex. The highest moderation levels have a very high detail grade when adding content.

Editing The editing task is similar to creation, it will return a formulary with the current data based on your permission level.

Deletion Usually in forums a moderator can delete user entries. In e-cidadania that is not the objective. User generated content must be preserved, it only can be deleted by platform administrators.

Move note This permission is for the debate system. IT allows or restricts the user capability of moving notes across the debate. The notes moving task is reserved for the space administrators.

Add/Remove

Groups

Groups are a massive way to give permissions to people. In this version groups will be a way to group people inside the spaces, except that for any reason you'll have to give them another permission for a specific task.

Adding/Removing groups

Adding people to groups

2.2.3 Spaces

What are they

How spaces work

2.2.4 Modules

Debates

Debate creation

Moderation

Proposals

Creating a proposal set

How to merge proposals

Voting

Creating a set

Moderation

2.2.5 Frequent errors

The most frequest errors are due to the server or a bad administrator management in the groups/permission case.

Appearance / Themes

3.1 e-cidadania themes

3.1.1 Other themes

This is a little index of the e-cidadania themes.

3.1.2 Create your own theme for e-cidadania

Creating a new theme for e-cidadania can be hard, but not impossible. We will give you the precise instructions here.

3.1.3 Reference

e-cidadania themes are mostly located in the proper modules. Every application manages its own themes, not regarding the style, but the distribution.

Apart from this, there are some general templates located at the "templates" directory.

General

Note: Still on development.

User profile

Note: Still on development.

Proposals

Note: Still on development.

Debates

Note: Still on development.

News

Note: Still on development.

Documents

Development

4.1 Creating the environment

For the people wanting to develop e-cidadania, we recommend to setup a new environment using buildout.

4.1.1 Steps

Below are the steps explained in a greater detail.

- Clone the official git repository of e-cidadania.

```
$ git clone https://github.com/cidadania/e-cidadania
```

- cd into the e-cidadania folder just cloned.

```
$ cd e-cidadania
```

- Just to be sure, when you do a ls, you should find a bootstrap.py python module and a buildout.cfg buildout configuration file.

```
~/e-cidadania$ ls
bootstrap.py  buildout.cfg  docs  __init__.py  README.rst  setup.py  src  tests
```

- Now run the bootstrap.py script with your system python.

```
~/e-cidadania$ python bootstrap.py
```

- If the bootstrapping procedure runs successfully, you will see the following output in the terminal.

```
~/e-cidadania$ python bootstrap.py
Downloading http://pypi.python.org/packages/2.7/s/setuptools/setuptools-0.6c11-py2.7.egg
Creating directory '/home/user/e-cidadania/bin'.
Creating directory '/home/user/e-cidadania/parts'.
Creating directory '/home/user/e-cidadania/eggs'.
Creating directory '/home/user/e-cidadania/develop-eggs'.
Getting distribution for 'setuptools'.
Got setuptools 0.6c12dev-r88846.
Generated script '/home/user/e-cidadania/bin/buildout'.
```

- Run the buildout script created in e-cidadania/bin

```
~/e-cidadania$ bin/buildout
```

Now buildout will run and download all the packages required for development in the e-cidadania.eggs folder and add them to the python path of the bin/django script which will be created after buildout runs successfully.

buildout will download the django framework itself and other packages as defined in buildout.cfg. bin/django is a wrapper around manage.py and works exactly the same way as manage.py and it must be used to run any django management commands while you are in the development environment.

> **Warning:** On some linux systems, running bin/buildout fails when installing PIL. The solution is to install python-imaging and in some environments python-devel too. Install python-imaging and re-run bin/buildout.

If the step above completes sucessfully, you will find a number of scripts in bin folder. Make sure that you have django and python in the bin folder.

- Now cd into the src folder.

  ```
  ~/e-cidadania$ cd src
  ```

- Generate the database:

  ```
  ~/e-cidadania/src$ ../bin/django syncdb
  ```

- Copy all the static files:

  ```
  ~/e-cidadania/src$ ../bin/django collectstatic
  ```

- Run the development server:

  ```
  ~/e-cidadania/src$ ../bin/django runserver
  ```

That's it!

4.1.2 Running the tests

We have drifted away from the Django way of keeping the tests with the source itself. All the tests are in a single folder called 'tests' which you can find at the root of e-cidadania. By default, the test runner is configured to run all the tests present in the test folder.

- To run all the tests:

  ```
  ~/e-cidadania$ bin/django test
  ```

or when you are in src:

```
~/e-cidadania/src$ ../bin/django test
```

- To run the tests with coverage:

  ```
  ~/e-cidadania$ bin/django test --with-cov
  ```

or when you are in src:

```
~/e-cidadania/src$ bin/django test --with-cov
```

4.2 Style guide

The style guide establish a series of rules to follow when coding in e-cidadania. These rules are unbreakable. The style guide follows closely the PEP8 document, with some exceptions provided from the internal style guide at Pocoo.

4.2.1 Python

Imports Every import must be situated in the file header, below to the comment header. The python imports must precede any others, and the external librearies or third party modules must precede the application ones.

Example:

```python
import os
import sys

from extlib import function

from myapp.module import function
```

Line width (columns) The code must be always 80 columns wide except on templates.

Long declarations If a code line does not fit in 80 columns, try to reduce it declaring variables previously. If it still can not fit, you can divide the lines this way:

Parentheses:

```python
website = models.URLField(_('Website'), verify_exists=True,
                          max_length=200, null=True, blank=True,
                          help_text=_('The URL will be checked'))
```

Declararations:

```python
this_is_a_very_long(function_call, 'with many parameters') \
    .that_returns_an_object_with_an_attribute

MyModel.query.filter(MyModel.scalar > 120) \
              .order_by(MyModel.name.desc()) \
              .limit(10)
```

Lists, tuples and dictionariess:

```python
items = [
    'this is the first', 'set of items', 'with more items',
    'to come in this line', 'like this'
]

dict = {
    ('mobile': phone),
    ('car': key),
    ('another': thing),
}
```

Indentation Indentation must be *always* 4 spaces per level, no exceptions. You can not use tabs for indenting.

Blank lines Every classes and method must be separated by two blank lines. The code inside a class or method by one blank line.

Example:

```python
class ListDocs(ListView):
    ----blank line----
    """
    List all documents stored whithin a space.
    """
    paginate_by = 25
    context_object_name = 'document_list'
    ----blank line----
    def get_queryset(self):
        place = get_object_or_404(Space, url=self.kwargs['space_name'])
        objects = Document.objects.all().filter(space=place.id).order_by('pub_date')
        return objects
```

```
    ----blank line----
    def get_context_data(self, **kwargs):
        context = super(ListDocs, self).get_context_data(**kwargs)
        context['get_place'] = get_object_or_404(Space, url=self.kwargs['space_name'])
        return context
----blank line----
----blank line----
def whatever(args):
    ----blank line----
    """
    A comment.
    """
    this_is_something = 0
```

4.2.2 HTML

Columns HTML code does not have a column limit, but it must be indented in a way we can locate easily every element inside the document. The indentation preceeds rendered results in application.

Indentation The X/HTML code must be indented with four spaces like python code, no exceptions.

Note: There may be some old code that follows the old indentation rule (2 spaces per level). If you see it we would be glad to receive a path to solve it.

4.2.3 CSS

Indentation Indentation will be 4 spaces, always, like Python code.

Example:

```
body {
    background: #FAFAFA;
        padding: 0;
        margin: 0;
        font-family: Verdana, "Lucida Sans", Arial;
        font-size: 1em;
        color: #000;
        cursor: default;
}
```

Colors Colors must be always written in hexadecimal. You are allowed to use three digits abbreviations.

Font size Font size must be declared in **em's** except for presentation requirement.

4.2.4 JavaScript

The JavaScript code follows the same rules from the python code.

4.3 User accounts

The user account system in e-cidadania is abased on the django *auth* module and in django-userprofile, created by James Bennet.

It might be that for other installations of e-cidadania you need other user data instead of the provided by the e-cidadania default installation. In that case you can modify the data model of the user.

The file containing the data model for the user profile is found in *apps/accounts/models.py*. All the fields can be replaced except *age* and *spaces*.

Note: Remember, after modifying the data model you will need to rebuild your database. We encourage you to have some application for database schema migration like *django-evolution*

Users are spearated in two parts. One of them is the django user account created by the *auth* module and the other is the *profile* object containing all the profile data for the user.

Users can be created without profile, but once you try to create a profile you will have to fill it and it will be linked to the user.

4.3.1 Public profiles

Users have a public zone in their profiles, which will show the data they want to show.

The public data will be visible to anyone visiting the profile, platform user or not.

Note: Be careful with the personal data you show on the net.

> **Warning:** Currently the data on the public profile is not configurable by the users. This is expected to change in e-cidadania 0.1.5.

4.4 Creating modules

e-cidadania is extensible through modules, which are nothing more than django apps.

> **Warning:** If you want to install modules, you will have to do it on deployment. Django does not support modules hotplugging.

4.4.1 Structure

A module is basically a django application which we integrate in e-cidadania. At this time we advocate for the django default structure in the distribution and file names.

> **Warning:** It's recommended to have expertise in django and python before creating a new module.

A module has three basic components, the data model, the view and the template.

Note: Discuss if is worthy to explain the creation of a module.

4.5 Generating the documentation

e-cidadania documentation is generated through sphinx (1.1.x) in three languages by default, which are:

- English
- Spanish
- Galician

The current structure of the documentation is:

```
docs/
    en/
    es/
    gl/
    reference/
    build/
    dev/ *
    docs/ *
    theming/ *
    index.rst *
    authors.rst *
```

Note: The folders and files marked with "*" are soft links to the english version of the documents.

The folders *en*, *es* and *gl* contain the documentation for each language. The *reference* folder is the autogenerated documentation from the code comments, which will always be in english and therefore is linked to all the languages to have just one copy of the reference instead of updating three.

We divide the documentation into four big categories:

```
dev/       -- Development or administration related (installation, configuration and deployment)
docs/      -- Usual documentation: user manual, administrator manual, etc.
theming/   -- Theming related tutorials, index of themes, etc.
reference/ -- Reference documentation generated from the code comments.
```

The documentation was made to be the most automated possible. The three languages are generated at the same time when *make html* is executed.

Once that command is executed the documentation will be in the *build* folder which is divided this way:

```
build/
    doctrees/
    html/
        es/
        en/
        gl/
    latex/
        es/
        en/
        gl/
```

The folders *es*, *en* and *gl* contain the documentation for their respective languages. The *doctrees* folder is just for reference when generating the documents.

As you can see, there are two formats of documentation: *latex* and *html*. e-cidadania has three officially supported formats for the documentation: html, latex and PDF (generated through pdflatex and stored in the *latex* directory).

Note: The way we designed the multilanguage system it should allow to build every format supported by sphinx (like epub, manpages, htmlhelp, etc.), but we don't guarantee it.

In the same way that you can build all the languages at once, you can also build them separately, executing the same *make* command but inside the language folder.

4.5.1 Linguistic rules

Not applicable to english translation at this time.

4.5.2 Fuzzy words

Not applicable.

4.6 Translations

The translation for the e-cidadania platform can be done in two ways:

- Transifex
- gettext

Both ways of translating are simple thanks to the Django *middleware*, a couple scripts and the transifex-client software.

Instead of making one global translation file, we decided to keep one translation file for each application, this way even if the modules change the translations will keep intact.

4.6.1 Translating with Transifex

To make translations with transifex you will need to have an account. After that you can visit the e-cidadania project page here.

After that you will need to sign up for a language team, and the administrator will have to approve it.

Here is what you will see:

image here

You will have to select a language and after that, the component that you want to translate. We split the components like the modules of e-cidadania, to maintain the application integrity.

Once you finished you don't have to do anything, just keep updating the translations until the string freeze period (usually 15 days before the release)
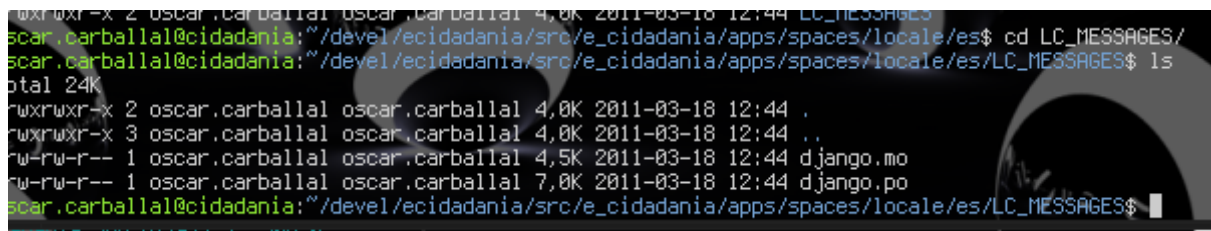

Note: Section incomplete.

4.6.2 Translating with gettext

Gettext is a well known tool by all the translators around the world. Its a standard. Thanks to the django *middleware* our work will be minimum, we only have to edit the .po files in the source files.

Instead of making one global translation, we decided to keep a translation file for every module, that way the translations will keep even if the modules are moved.

The location of the strings is usually a directory called **locale** inside the module. Inside it, you can find directories with the country code (en, es, us, gl, fr, etc.) and inside this one, the PO and MO files.

To translate, you must edit the PO file, which is a plain text file.

The MO file is the compiled version of the translation so the machine can read it to use it.

Once you edited the .po, you have to send a pull request to the repository, and that's all.

> **Warning:** Stablish a workflow for translators and explain it here.

Reference

5.1 `apps.ecidadania.cal` — Calendar and events

5.1.1 `cal.models` — Data models

The calendar module calls a version of Python HTML Calendar and adds some functions to use django objects with it.

The source code is based on the work of Eivind Uggedal <eivind@uggedal.com>

class apps.ecidadania.cal.models.**EventCalendar**(*LocaleHTMLCalendar*)

Event calendar is a basic calendar made with HTMLCalendar module and its instance LocaleHTMLCalendar for translation.

> **Attributes** LocaleHTMLCalendar
>
> **Methods** formatday, formatmonth, group_by_day, day_cell

day_cell(*cssclass*, *body*)

Create the day cell.

formatday(*day*, *weekday*)

Format the day cell with the current events for the day.

formatmonth(*year*, *month*)

Format the current month wuth the events.

group_by_day(*events*)

Group the returned events into their respective dates.

5.1.2 `cal.views` — Views

class apps.ecidadania.cal.views.**calendar**(*request*, *space_name*, *year*, *month*)

Returns an localized event calendar with all the Meeting objects.

> **Context** calendar, nextmonth, prevmonth, get_place
>
> **Returns** Localized HTML Calendar

5.2 `apps.ecidadania.debate` — Debates

5.2.1 `debate.admin` — Administration

5.2.2 `debate.models` — Data models

This file contains all the data models for the debate module.

class `apps.ecidadania.debate.models.`**`Debate`**(*models.Model*)
> Debate object. In every space there can be unlimited debates, each one of them holds all the related notes. Debates are filtered by space. Start/End dates are for letting users use the debate or not. New in version 0.1b.

class `apps.ecidadania.debate.models.`**`Note`**(*models.Model*)
> The most important object in every debate, the message. It has a coordinates value to determine the position of the note in its debate. New in version 0.1b.

5.2.3 `debate.forms` — Forms

This file contains all the forms for the debate modules.

class `apps.ecidadania.debate.forms.`**`DebateForm`**(*ModelForm*)
> Returns an empty form for creating a new Debate.

> > **Return type** HTML Form

> New in version 0.1b.

class `apps.ecidadania.debate.forms.`**`NoteForm`**(*ModelForm*)
> Returns an HTML Form to create or edit a new 'note' or 'proposal' like it's called on the sociologists argot.

> > **Return type** HTML Form

> New in version 0.1b.

5.2.4 `debate.views` — Views

These are the views that control the debates.

`apps.ecidadania.debate.views.`**`add_new_debate`**(*request*, *space_name*)
> Create a new debate. This function returns two forms to create a complete debate, debate form and phases formset. New in version 0.1.5.

> > **Attributes** debate_form, row_formset, column_formset

> > **Context** form, rowform, colform, get_place, debateid

`apps.ecidadania.debate.views.`**`get_debates`**(*request*)
> Get all debates and serve them through JSON.

`apps.ecidadania.debate.views.`**`create_note`**(*request*, *space_name*)
> This function creates a new note inside the debate board. It receives the order from the createNote() AJAX function. To create the note first we create the note in the DB, and if successful we return some of its parameters to the debate board for the user. In case the petition had errors, we return the error message that will be shown by jsnotify. New in version 0.1.5.

`apps.ecidadania.debate.views.`**`update_note`**(*request*, *space_name*)
> Updated the current note with the POST data. UpdateNoteForm is an incomplete form that doesn't handle some properties, only the important for the note editing.

`apps.ecidadania.debate.views.`**`delete_note`**(*request*, *space_name*)
> Deletes a note object.

class `apps.ecidadania.debate.views.`**`ViewDebate`**(*DetailView*)
    View a debate.

>   **Context** get_place, notes, columns, rows

class `apps.ecidadania.debate.views.`**`ListDebates`**(*ListView*)
    Return a list of debates for the current space.

>   **Context** get_place

5.3 `core.spaces` — Spaces/Processes

5.3.1 `spaces.admin` — Administration

e-cidadania administration models for django-admin. This administration models will make their respective data models available for management.

class `core.spaces.admin.`**`EntityAdmin`**(*admin.ModelAdmin*)
    Entities administration model.

>   **List fields** name, website, space

>   **Search fields** name

class `core.spaces.admin.`**`EntityInline`**(*admin.TabularInline*)
    TabularInline view for entities.

>   **`model`**
>   alias of `Entity`

class `core.spaces.admin.`**`SpaceAdmin`**(*admin.ModelAdmin*)
    Administration view for django admin to create spaces. The save() method is overriden to store automatically the author of the space.

>   **List fields** name, description, date

>   **Search fields** name

class `core.spaces.admin.`**`DocumentAdmin`**(*admin.ModelAdmin*)
    Administration view to upload/modify documents. The save() method is overriden to store the author automatically.

>   **List fields** title, space, docfile, author, pub_date

>   **Search fields** title, space, author, pub_date

class `core.spaces.admin.`**`EventAdmin`**(*admin.ModelAdmin*)
    Meetings administration model.

>   **List fields** title, space, meeting_date

>   **Search fields** title

5.3.2 `spaces.models` — Data models

class `core.spaces.models.`**`Space`**(*models.Model*)
    Spaces model. This model stores a "space" or "place" also known as a participative process in reality. Every place has a minimum set of settings for customization.

    There are three main permission roles in every space: administrator (admins), moderators (mods) and regular users (users).

class `core.spaces.models.`**`Entity`**(*models.Model*)
    This model stores the name of the entities responsible for the creation of the space or supporting it.

class `core.spaces.models.`**`Document`** (*models.Model*)
> This models stores documents for the space, like a document repository, There is no restriction in what a user can upload to the space.
>
>> **Methods**   get_file_ext, get_file_size

class `core.spaces.models.`**`Event`** (*models.Model*)
> Meeting data model. Every space (process) has N meetings. This will keep record of the assistants, meeting name, etc.

5.3.3 `spaces.forms` — Forms

This module contains all the space related forms, including the forms for documents, meetings and entities. Most of the forms are directly generated from the data models.

class `core.spaces.forms.`**`SpaceForm`** (*ModelForm*)
> Returns a form to create or edit a space. SpaceForm inherits all the fields from the `Space` data model.
>
>> **Return type**   HTML Form
>
> New in version 0.1.

class `core.spaces.forms.`**`DocForm`** (*ModelForm*)
> Returns a form to create or edit a space related document, based on the spaces.Document data model.
>
>> **Return type**   HTML Form
>
> New in version 0.1.

class `core.spaces.forms.`**`EventForm`** (*ModelForm*)
> Returns a form to create or edit a space related meeting, based on the spaces.Meeting data model.
>
>> **Return type**   HTML Form
>
> New in version 0.1.

`EntityFormSet` — modelformset_factory

5.3.4 `spaces.views` — Views

General spaces views

Spaces views

Document views

Meeting views

5.4 `apps.ecidadania.proposals` — Proposals

5.4.1 `proposals.admin` — Administration

The proposal administration allows to edit every proposal made in the system.

class `apps.ecidadania.proposals.admin.`**`ProposalAdmin`** (*admin.ModelAdmin*)
> Basic proposal administration interface since most of the work is done in the website.
>
>> **List display**   title, author, tags
>
>> **Search**   none:

5.4.2 `proposals.models` — Data models

Proposal data models are the ones to store the data inside the DB.

class `apps.ecidadania.proposals.models.`**`Category`**(*BaseClass*)
 Dummy class for proposal categories. Inherits directly from `BaseClass` without adding any fields.

class `apps.ecidadania.proposals.models.`**`Proposal`**(*models.Model*)
 Proposal data model. This will store the user proposal in a similar way that Stackoverflow does. Take in mind that this data model is very exhaustive because it covers the administrator and the user.

> **Automatically filled fields** Space, Author, Pub_date, mod_date.
>
> **User filled fields** Title, Description, Tags, Latitude, Longitude.
>
> **Admin fields (manual)** Code, Closed, Close_reason, Anon_allowed, Refurbished, Budget.
>
> **Admin fields (auto)** Closed_by
>
> **Extra permissions** proposal_view

 `CLOSE_REASONS` for :class:Proposal data model is hardcoded with four values, which will fit most of the requirements.

5.4.3 `proposals.forms` — Forms

Proposal forms.

class `apps.ecidadania.proposals.forms.`**`ProposalForm`**(*ModelForm*)
 ProposalForm is a basic form autogenerated form for Proposal model. Returns an empty form for creating a new proposal.

> **Return type** HTML Form

 New in version 0.1.5b.

5.4.4 `proposals.views` — Views

5.5 `apps.ecidadania.news` — News

5.5.1 `news.admin` — Administration

class `apps.ecidadania.news.admin.`**`PostAdmin`**(*admin.ModelAdmin*)
 Administration view for news.

5.5.2 `news.models` — Data models

class `apps.ecidadania.news.models.`**`Post`**(*models.Model*)
 Model of a news post.

5.5.3 `news.forms` — Forms

class `apps.ecidadania.news.forms.`**`NewsForm`**(*ModelForm*)

5.5.4 `news.views` — Views

class `apps.ecidadania.news.views.`**`ViewPost`**(*DetailView*)

    View a specific post.

    `get_context_data`(***kwargs*)

        Get extra context data for the ViewPost view.

class `apps.ecidadania.news.views.`**`AddPost`**(*FormView*)

    Create a new post. Only registered users belonging to a concrete group are allowed to create news. only site administrators will be able to post news in the index page.

        Parameters   space_url

        Context   get_place

class `apps.ecidadania.news.views.`**`EditPost`**(*UpdateView*)

    Edit an existent post.

        Parameters   space_url, post_id

        Context   get_place

class `apps.ecidadania.news.views.`**`DeletePost`**(*DeleteView*)

    Delete an existent post. Post deletion is only reserved to spaces administrators or site admins.

    `get_context_data`(***kwargs*)

        Get extra context data for the ViewPost view.

Getting help

- Try looking the FAQ.
- e-cidadania mailing lists
    - ecidadania-users@freelists.org
    - ecidadania-dev@freelists.org
    - ecidadania-es@freelists.org
- Bug reporting in the e-cidadania bug tracker or contact us through e-mail at soporte@ecidadania.org

```
authors
```

Python Module Index

a

c