Making Arches Multilingual: a guidebook (Version 1)

Andrea Zerbini (andrea.zerbini@arch.ox.ac.uk)

The Endangered Archaeology in the Middle East and North Africa (EAMENA) project based at the University of Oxford has been customising Arches 3 with the aim of recording cultural heritage and damage to it across the MENA region.
In a region composed of 20 countries, all but one (Iran) primarily Arabic speaking, full multilingual support in English and Arabic was considered a priority for the team.

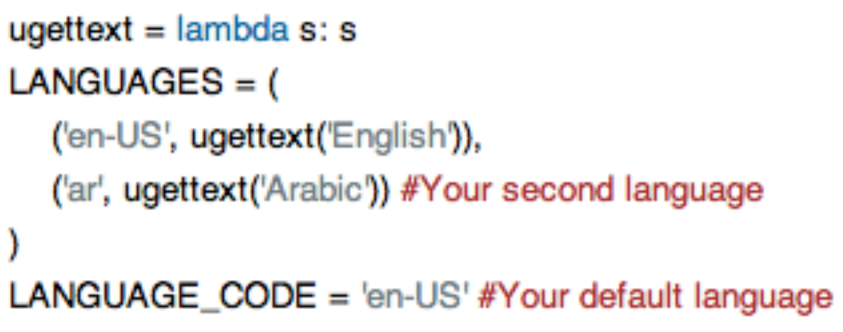
Translation involves both the static strings located in the Arches and Django python files (as well as in the html templates and javascript libraries) and the concept labels and notes saved in the Arches Resource Data Manager (RDM). The former can be easily achieved by following the steps provided by the Django online documentation on this matter (https://docs.djangoproject.com/en/1.9/topics/i18n/translation/).

The latter requires a number of additional steps as illustrated in this guidebook.

Step 0. Preliminary changes to settings.py

These changes should be applied when activating Django's internationalization.

- Uncomment 'django.middleware.locale.LocaleMiddlewareyou're your MIDDLEWARE_CLASSES in arches/settings.py
- In your app's settings.py, add the following:

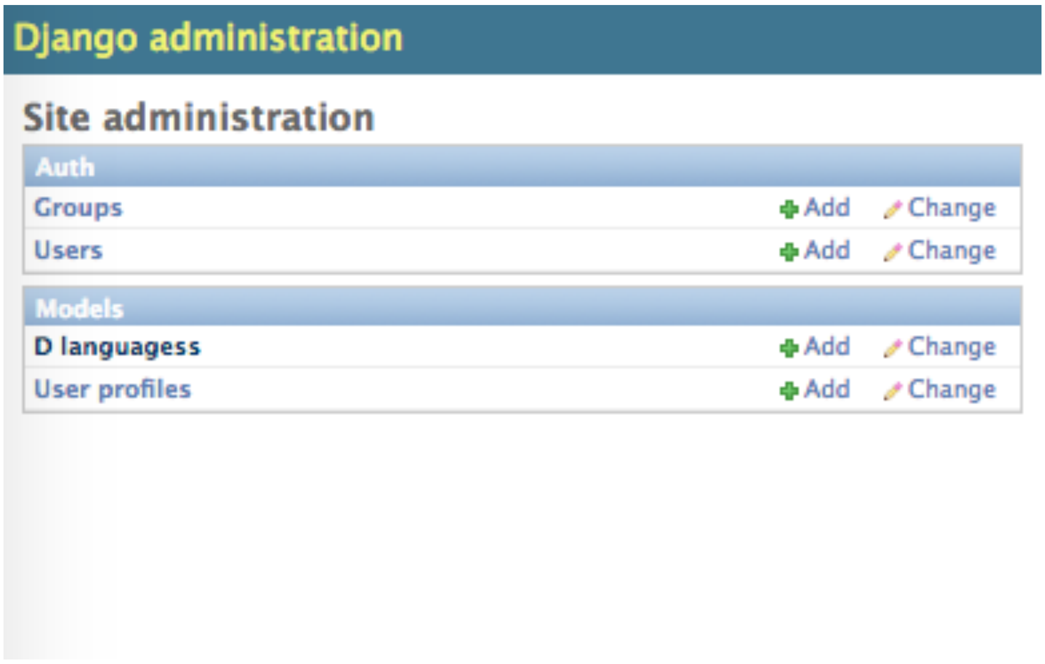


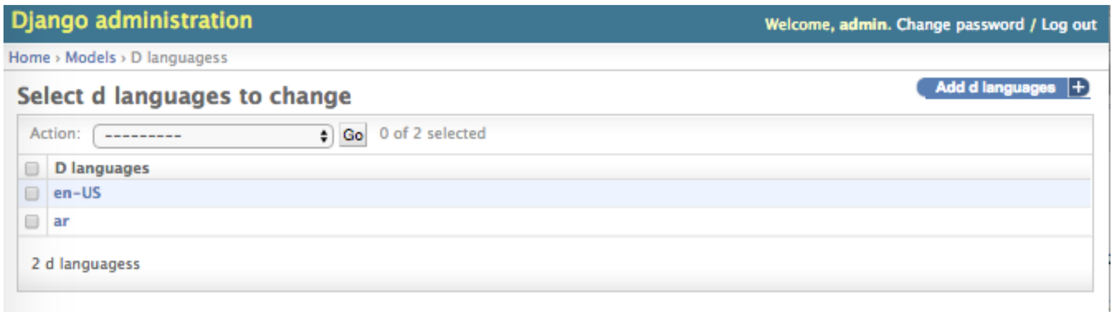
```
ugettext = lambda s: s
LANGUAGES = (
    ('en-US', ugettext('English')),
    ('ar', ugettext('Arabic')) #Your second language
)
LANGUAGE_CODE = 'en-US' #Your default language
```

(explanation in http://www.djangobook.com/en/2.0/chapter19.html)

Step 1. Adding a second language support in your Django admin panel

Navigate to your Django admin panel (yourwebsite.com/admin or localhost:8000/admin if you are offline). Arches by default links the model d_languages (a table in the Reference data section of the database) to the admin panel (you can add further models to the admin panel by changing arches/admin.py). Click on d_languages and add a new language by setting Languageid = your second language's standard ISO 639-1 language code (e.g. for standard Arabic this is 'ar').




Django administration
Welcome, admin. Change password / Log out
Home › Models › D languagess
Add d languages
Action:

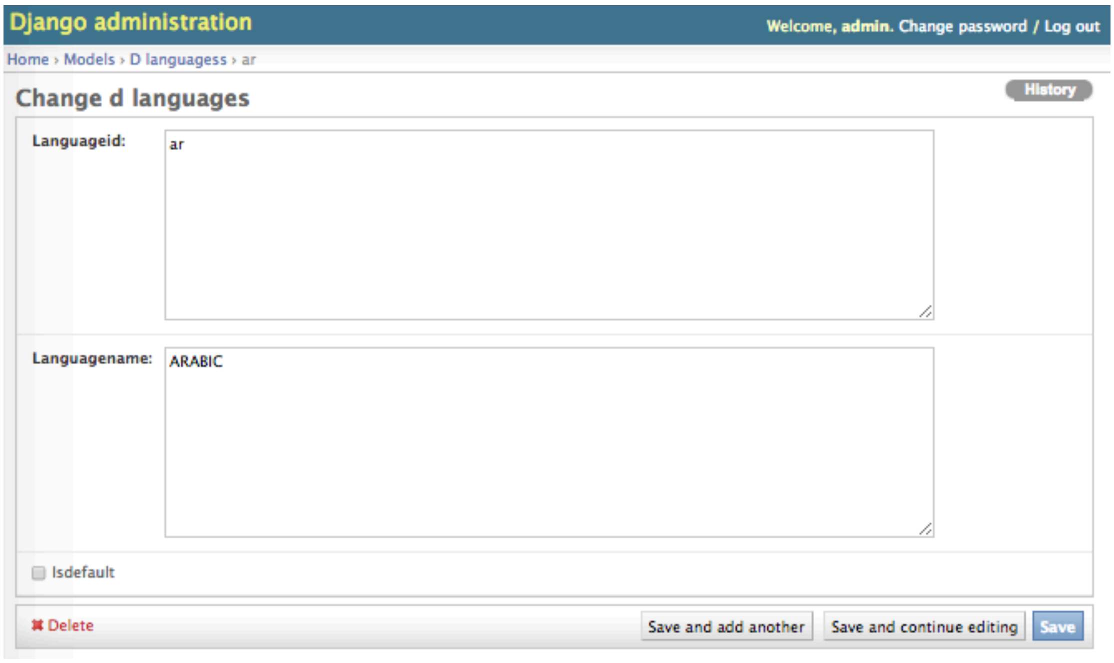
Go

Select d languages to change

0 of 2 selected

D languages
en-US
ar

2 d languagess


Django administration
Welcome, admin. Change password / Log out
Home › Models › D languagess › ar
History
Delete
Save and add another
Save and continue editing
Save

Change d languages

Languageid: ar

Languagename: ARABIC

☐ Isdefault

Step 2. Modifying your header.htm template

You are now ready to start modifying the Arches files to support the translation of concept labels. First of all, you will have to modify your header to dynamically obtain a list of languages from the database table d_languages:

- Navigate to your app's template folder and open header.htm, then navigate to <a>{% trans "Languages" %}</a> Replace with the following:

```
<a>{% trans "Languages" %}</a>
<form action="{% url 'set_language' %}" method="post">
{% csrf_token %}
  <input name="next" type="hidden" value="{{ request.get_full_path }}" />
  <select name="language">

<ul class="lenguages">
{% for language in language_list %}
    <li class="active">
        <option value="{{ language.languageid }}"{% if language.languageid == LANGUAGE_CODE %} selected="selected"{% endif %}><a
        href="javascript:void(0);">{{ language.languagename }}<i class="fa fa-check"></i></a> </option>
    </li>
{% endfor %}
  </select>
<input type="submit" value="Change" />
</form>
```

(I have yet to improve on the visualisation of the header. Ultimately I will use a KO viewmodel for this)

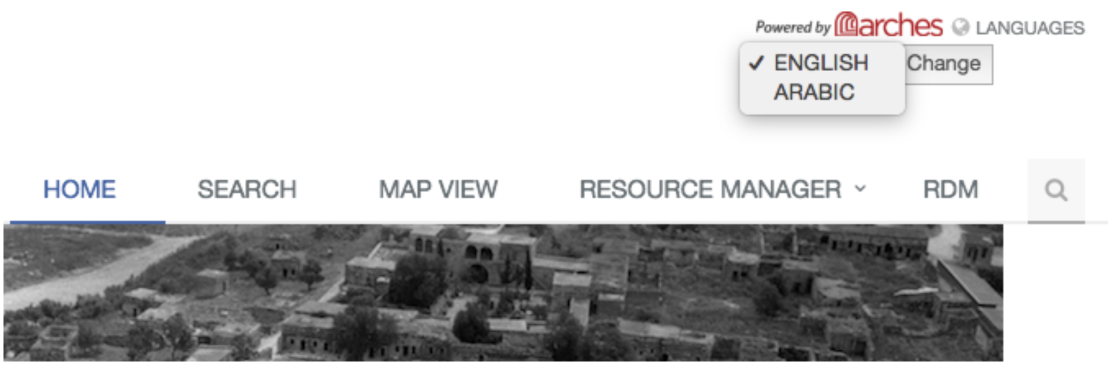
Step 3. Adding the header context processor and the i18n url

In order to pull the languages from the database, a custom context processor is needed. Navigate to /arches/app/utils/context_processors.py and add the following:

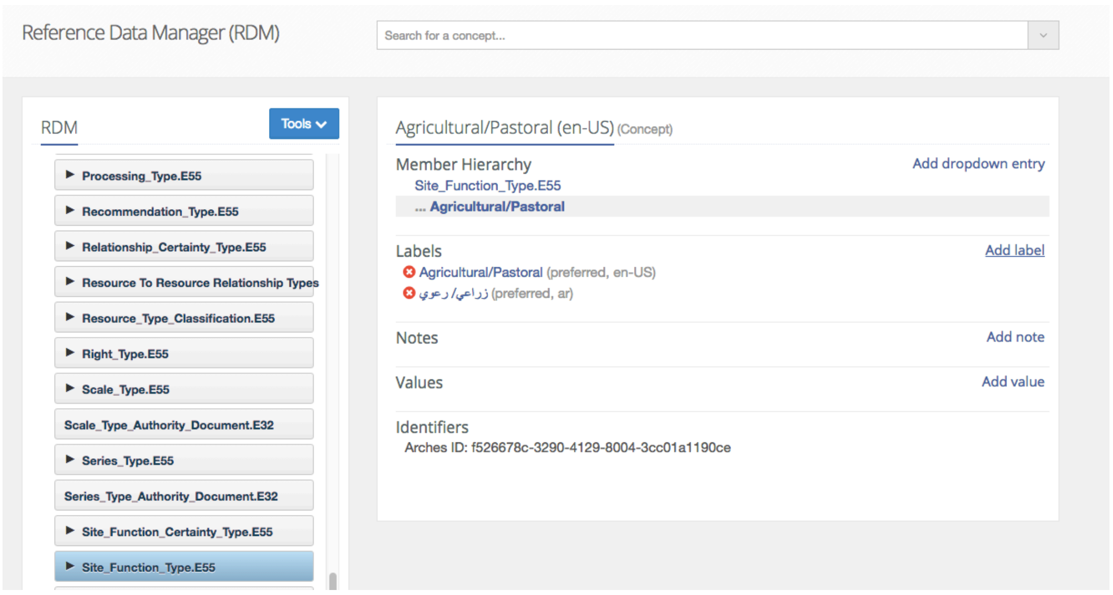
```
def header(request):
    languages = DLanguages.objects.all()
    return {
        'language_list': languages
    }
```

Then navigate to arches/urls.py and add, at the bottom of urlpatterns, the following line: url(r'^i18n/', include('django.conf.urls.i18n')),
See here (https://docs.djangoproject.com/en/1.9/topics/i18n/translation/#the-set-language-redirect-view) for an explanation of this step.

At this point, you should be able to visualise a dropdown containing the languages stored in the DB table d_languages:



You should now proceed to enter your second language's concept labels in the Arches' RDM, see example below:


Reference Data Manager (RDM)

Search for a concept...

RDM — Tools

- Processing_Type.E55
- Recommendation_Type.E55
- Relationship_Certainty_Type.E55
- Resource To Resource Relationship Types
- Resource_Type_Classification.E55
- Right_Type.E55
- Scale_Type.E55
- Scale_Type_Authority_Document.E32
- Series_Type.E55
- Series_Type_Authority_Document.E32
- Site_Function_Certainty_Type.E55
- Site_Function_Type.E55

Agricultural/Pastoral (en-US) (Concept)

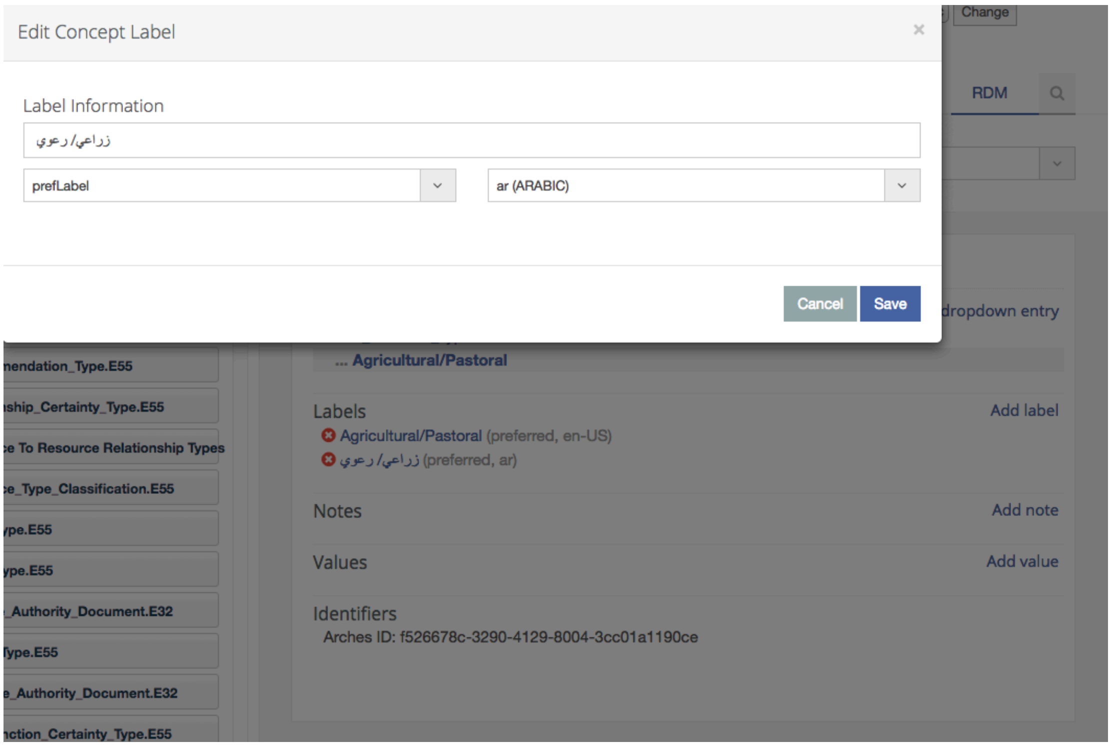
Member Hierarchy — Add dropdown entry
Site_Function_Type.E55
... Agricultural/Pastoral

Labels — Add label
Agricultural/Pastoral (preferred, en-US)
زراعي/رعوي (preferred, ar)

Notes — Add note

Values — Add value

Identifiers
Arches ID: f526678c-3290-4129-8004-3cc01a1190ce


Edit Concept Label

Label Information
زراعي/رعوي
prefLabel — ar (ARABIC)

Cancel — Save

Once you have fully translated all of your concepts, you can move on to the next steps. The next steps illustrate how to make sure that, once a second language is selected (that is, a language that is different from the default language given in settings.LANGUAGE_CODE), the concept labels in the Report page, Dropdowns, and the selected branches in the Resource Editor are all displayed in this language.

Step 4. Report page changes

In your Arches_HIP /views folder, navigate to resources.py. In the report() method, you will need to replace lang = settings.LANGUAGE_CODE with lang=

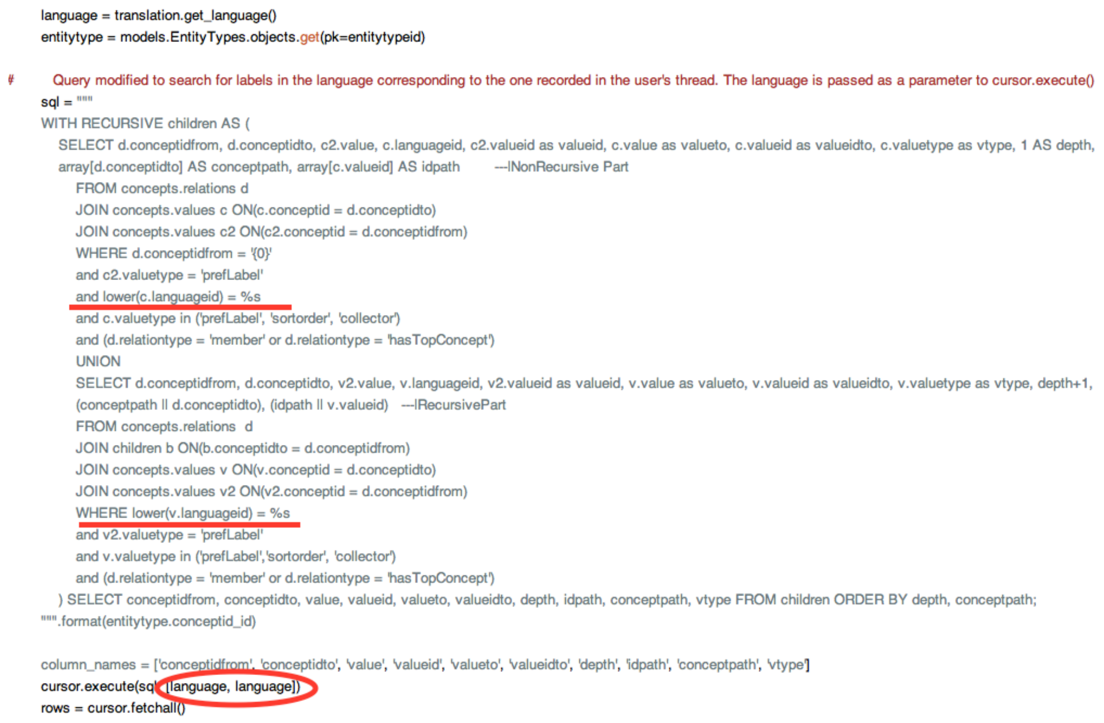
request.GET.get('lang', request.LANGUAGE_CODE) . This will allow the method to obtain the language code from Django's set_language() (called by the header form) instead of by looking it up in the settings.py file. The Report page will now correctly display in the language selected via header.htm.

Step 5. Populating the Dropdowns in the correct language in Editing mode

Dropdowns (E55 nodes) are populated via the method get_e55_domain() which gets called by your app's models/forms.py. That method is defined in arches/app/models/concept.py to which you will now need to navigate. Navigate to the get_e55_domain() method (should be around l. 620). By default, the selection query 'sql' does not filter by languageid. As a result, prefLabels get selected in a weird alphabetical order **across** multiple languages, and you will end up with some labels in a language, and some in another. To avoid this, you should pass the currently selected language to get_e55_domain() and then modify 'sql' so as to have a WHERE clause that filters by language.

First, at the top of concepts.py after the other imports, add from django.utils import translation. Then get back to get_e55_domain() and add language = translation.get_language() (see here for this method: https://docs.djangoproject.com/en/1.9/ref/utils/#django.utils.translation.get_language)

Then modify your 'sql' query as follows, and pass the language parameter to cursor.execute():



```
    language = translation.get_language()
    entitytype = models.EntityTypes.objects.get(pk=entitytypeid)

#       Query modified to search for labels in the language corresponding to the one recorded in the user's thread. The language is passed as a parameter to cursor.execute()
    sql = """
    WITH RECURSIVE children AS (
        SELECT d.conceptidfrom, d.conceptidto, c2.value, c.languageid, c2.valueid as valueid, c.value as valueto, c.valueid as valueidto, c.valuetype as vtype, 1 AS depth,
        array[d.conceptidto] AS conceptpath, array[c.valueid] AS idpath        ---|NonRecursive Part
            FROM concepts.relations d
            JOIN concepts.values c ON(c.conceptid = d.conceptidto)
            JOIN concepts.values c2 ON(c2.conceptid = d.conceptidfrom)
            WHERE d.conceptidfrom = '{0}'
            and c2.valuetype = 'prefLabel'
            and lower(c.languageid) = %s
            and c.valuetype in ('prefLabel', 'sortorder', 'collector')
            and (d.relationtype = 'member' or d.relationtype = 'hasTopConcept')
            UNION
            SELECT d.conceptidfrom, d.conceptidto, v2.value, v.languageid, v2.valueid as valueid, v.value as valueto, v.valueid as valueidto, v.valuetype as vtype, depth+1,
            (conceptpath || d.conceptidto), (idpath || v.valueid)   ---|RecursivePart
            FROM concepts.relations  d
            JOIN children b ON(b.conceptidto = d.conceptidfrom)
            JOIN concepts.values v ON(v.conceptid = d.conceptidto)
            JOIN concepts.values v2 ON(v2.conceptid = d.conceptidfrom)
            WHERE lower(v.languageid) = %s
            and v2.valuetype = 'prefLabel'
            and v.valuetype in ('prefLabel','sortorder', 'collector')
            and (d.relationtype = 'member' or d.relationtype = 'hasTopConcept')
        ) SELECT conceptidfrom, conceptidto, value, valueid, valueto, valueidto, depth, idpath, conceptpath, vtype FROM children ORDER BY depth, conceptpath;
    """.format(entitytype.conceptid_id)

    column_names = ['conceptidfrom', 'conceptidto', 'value', 'valueid', 'valueto', 'valueidto', 'depth', 'idpath', 'conceptpath', 'vtype']
    cursor.execute(sql, [language, language])
    rows = cursor.fetchall()
```

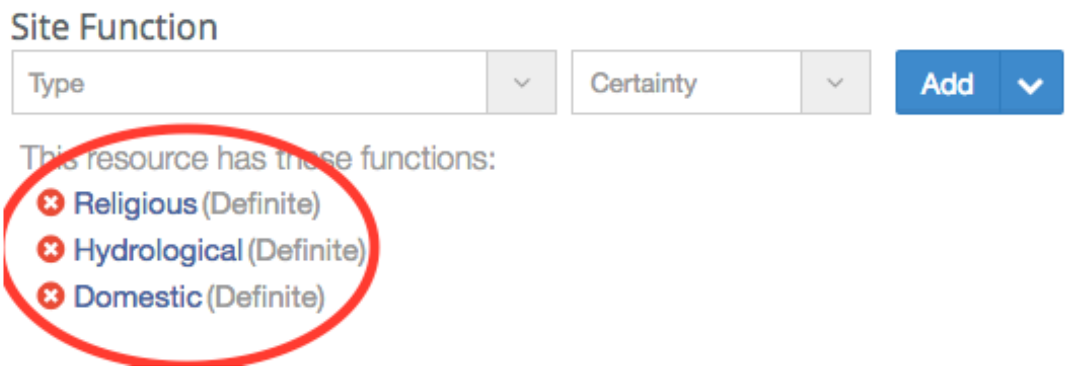
You will now be able to display Dropdown concept labels in the correct language. However, since there is a chance you may forget/decide that you do not want to translate all the concept labels, I suggest adding the following clause just below the above:

```
#       If a prefLabel in the right language does not exist, the query is defaulted to the language passed in settings.LANGUAGE_CODE
    if len(rows) == 0:
        cursor.execute(sql, [settings.LANGUAGE_CODE.lower(), settings.LANGUAGE_CODE.lower()])
        rows = cursor.fetchall()
```

If no label in the selected language is found, the method defaults to the default language as given in settings.py's LANGUAGE_CODE.

Step 6. Selected branches

Changes will be required also to the method get_nodes() (which may be found in arches/app/models/forms.py), which is responsible for populating the branches of a node (and its children) which are already stored in the database (e.g. see three branches of the node Site Function and its child (Site Function Certainty Type) below):



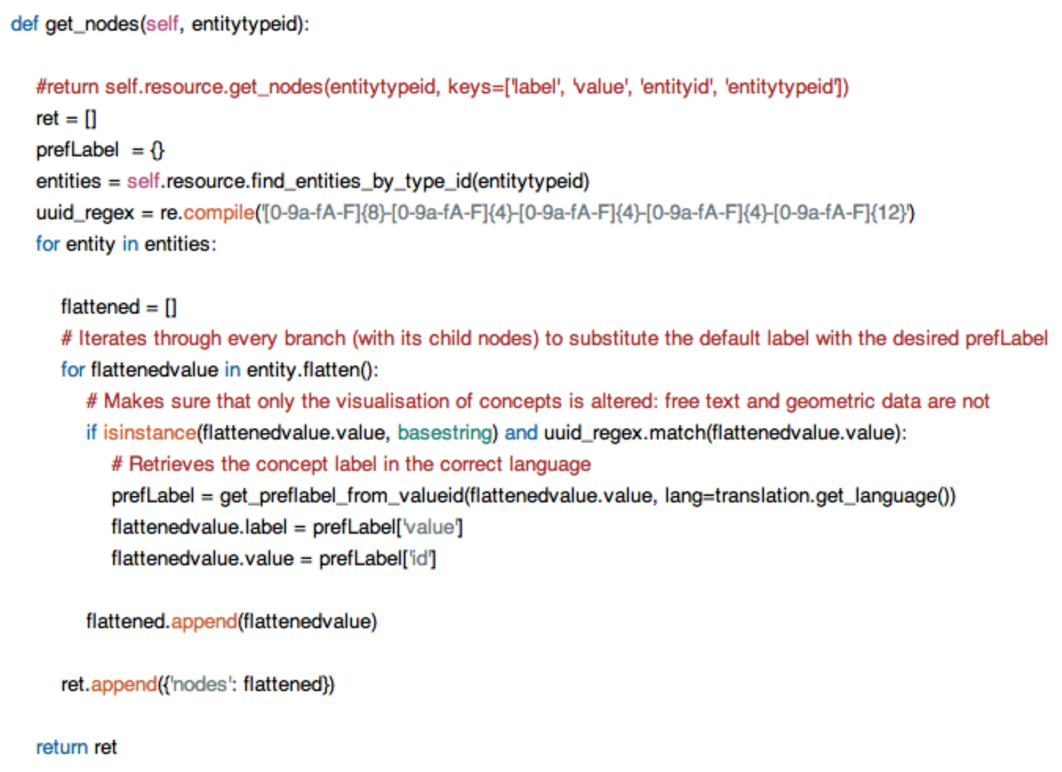
Site Function

Type | Certainty | Add

This resource has these functions:

- Religious (Definite)
- Hydrological (Definite)
- Domestic (Definite)

Navigate to arches/app/models/forms.py. Add 'import uuid' and 'from django.utils import translation' to the list of imports at the top. Then proceed to modify get_nodes as follows:



```
def get_nodes(self, entitytypeid):

    #return self.resource.get_nodes(entitytypeid, keys=['label', 'value', 'entityid', 'entitytypeid'])
    ret = []
    prefLabel  = {}
    entities = self.resource.find_entities_by_type_id(entitytypeid)
    uuid_regex = re.compile('[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}')
    for entity in entities:

        flattened = []
        # Iterates through every branch (with its child nodes) to substitute the default label with the desired prefLabel
        for flattenedvalue in entity.flatten():
            # Makes sure that only the visualisation of concepts is altered: free text and geometric data are not
            if isinstance(flattenedvalue.value, basestring) and uuid_regex.match(flattenedvalue.value):
                # Retrieves the concept label in the correct language
                prefLabel = get_preflabel_from_valueid(flattenedvalue.value, lang=translation.get_language())
                flattenedvalue.label = prefLabel['value']
                flattenedvalue.value = prefLabel['id']

            flattened.append(flattenedvalue)

        ret.append({'nodes': flattened})

    return ret
```

This method substitutes a concept's label (a string such as 'Building') and its value (a UUID) with the label and value corresponding to the concept in the selected language.

Step 7. Replacing settings.LANGUAGE_CODE with request.LANGUAGE_CODE in -in arches/app/views/resources.py and concept.py

This step may not be necessary, but I have replaced the standard settings.LANGUAGE_CODE throughout in arches/app/views/resources.py and concept.py with request.LANGUAGE_CODE. The only exception is the last if clause of get_preflabel_from_conceptid(), where if 'preflabel['_source']['language'] == settings.LANGUAGE_CODE and ret == None:' should be replaced with if preflabel['_source']['language'] == lang and ret == None:

That's it! Let me know if it works.

AZ