Preflib 2.0

Simon Rey, Nick Mattei

August 26, 2021

Contents

Chapter 1

Developing PrefLib

In this chapter, we detail the inner structure of PrefLib. We will first focus on the folder structure, explaining the role of each file. The second part of this chapter is devoted to the structure of the database.

The aim of this chapter is to provide all the necessary information to someone who would like to develop further the website. If you are only interested in maintaining the website and do some small changes, the second chapter might be more interesting to you.

1.1 Folder Structure

The folder structure of the project follows that of a typical Django project with one application. The overall Django project is called `preflib` and the main, and only, django application is called `preflibApp`. The overall folder structure can be found in Figure 1.1. In what follows, we give an explanation for each folder and files.

🗀 preflib

The `preflib` folder contains the files that have an impact on the entire project. This is the highest folder in the Django hierarchy. Files in there are mainly used to set up the global parameters of the project.

🗋 **settings.py** This is the main files for the global settings. Among other things, you will find there the settings for the database, the location of the static files, the debugging mode, the installed applications, ... Not that this file is not on the git for security reasons.

🗋 **urls.py** Use this file to set up the global rules for urls. Whenever a request passed over to Django, this file is used to decide where to send the request next. Handlers for the errors (404, 500, etc...) are also defined there.

🗋 **wsgi.py** This is only used to set up the connection between Django and whatever WSGI tool is used (uWSGI, Gunicorn, Passenger). If you did not understand the previous sentence, you will most likely never have to deal with this file.

🗀 preflibApp/management/commands

The `management/commands` folder includes the user-defined commands that one can access using the Django's management tool. Each file in this folder contains a class `Command` which should contain some specific methods. Importantly, once defined and put in this folder, the command can be accessed as any other management command, for instance from the command line using: `python3 manage.py adddataset`.

- preflib
  - settings.py
  - urls.py
  - wsgi.py
- preflibApp
  - management
    - commands
      - adddataset.py
      - generatezip.py
      - initializedb.py
      - updatemetadata.py
      - updatepapers.py
  - preflibtools
    - drawing.py
    - graph.py
    - instance.py
    - properties.py
  - static
    - adminLogo
      - images for the admin menu
    - css
      - nouislider.css
      - style.css
    - data
      - CD
      - ED
      - MD
      - types
    - datatoadd
    - js
      - nouislider.js
    - types
      - dataTypesImages.tex
      - images to present the types
    - crisner.png
    - gitHubLogo.png
    - papers.bib
    - preflibADTCite.txt
    - preflibADTPaper.pdf
    - preflibTrendsComSoCChapter.pdf
    - preflibTrendsComSoCCite.txt
  - templates
    - includes
      - HTML files that can be included
    - preflib
      - main HTML files
    - 400.html
    - 403.html
    - 404.html
    - 500.html
  - templatetags
    - extratags.py
  - admin.py
  - apps.py
  - choices.py
  - forms.py
  - models.py
  - scripts.py
  - urls.py
  - views.py

Figure 1.1: Folder Structure of the Project, some less relevant files have been omitted.

adddataset.py This command is used to add datasets to the database. Only zip files located in the folder `static/datatoadd` can be retrieved by this command. Two arguments can be passed to the command, either `--file zipfilename` to add only a specific zip file, or `--all` to add all the zip files in the `static/datatoadd` folder to the database.

When adding a dataset, the command unzip the archive in a temporary folder. It then goes through the files to find the info file used to get all the information for the dataset. Then, each file is added to the database, together with its datapatch (if needed). Note that the metadata are not computed when adding a dataset.

generatezip.py This command is used to generate all the zip files served by the website. It first generates the zip files for the dataset, creating the info file with the entry in the database. In a second time, it generates the zip files per type of data.

initializedb.py This command is only run once, at the very beginning, to populate the database with the entries that are needed. It mainly sets up the metadata.

updatemetada.py This command computes the metadata for the data files. Two options can be passed to the command. When used with `--dataset datasetAbbreviation`, only the metadata for the given dataset will be computed. One can also use `--noDrawing` to avoid generating the images for the data file (which takes a lot of time).

For each relevant data file, the command will go through all the *active* metadata (see later what active mean here). Whenever an active metadata applies to the data type of the data file, the corresponding function is called to compute its value. If drawing is allowed, the relevant drawing method is also called.

updatepapers.py This command updates the list of papers that are using Preflib. It reads the bib file `static/papers.bib` and update the database accordingly.

preflibApp/preflibtools

This folder contains the tools that were developed around PrefLib to work with the data. Note that this version is far from what you can find in the PrefLibTools repository on GitHub. These tools are mainly used to analyze the data when computing the relevant metadata.

drawing.py This file contains all the functions that are used to draw the images representing the data.

graph.py This file contains the definition of the `Graph` class that we use to represent graphs and access several useful methods for them.

instance.py This file contains the class used to represent PrefLib instances. The methods for parsing the data file are defined there.

properties.py This file contains a collection of functions used to check whether some properties hold for a given instance. This is the main file used to compute the metadata of the instances.

preflibApp/static

The `static` folder contains all the static files that are used (unsurprisingly). This folder is the one considered by the `collecstatic` management command from Django.

adminLogo This folder gathers all the images of the logo used for the admin menu. Note that it might disappear in some later versions.

css This folder gathers all the css files. There are currently two of them: `style.css`

js

types

crisner.png

gitHubLogo.png

papers.bib

preflib citations

preflibApp/templates

preflibApp/templatetags

preflibApp

- preflib
  - settings.py: all the inner settings of the project, only change things there if you know what you're doing.
  - urls.py: general patterns for URL, since there only is one application the file is quite basic. Handlers for errors (500, 404, ...) also are defined there.
  - wsgi.py: used to link django to the http server, do not modify.
- preflibApp
  - management/commands
    - adddataset.py: management command used to add a dataset to the database.
    - createmetadata.py: command to generate all the metadata for the datafile in the database.
    - generatezip.py: generates all the zip files.
    - initializedb.py: command to be run when setting up the website.
    - updatepapers.py: update the list of the papers using Preflib according the a bib file.
  - migrations: inner Django stuff, do not modify.
  - preflibtools: nothing to do with Django, all the tools used to deal with the data.
  - static: all the static files that are served by the website.
  - templates: a template is an html file which can incorporate some Django code to perform some computations in it. This folder contains the general html structure of the website, i.e., all its templates.
  - templatetags: custom tags that can be used in the templates.
  - admin.py: define which tables can be accessed on the Django admin page of the website.
  - apps.py: inner Django stuff, do not modify.
  - choices.py: several fields in the database have to be selected among a specific list. All the lists that are not worth putting in the database (because they never change for instance) are defined here.
  - forms.py: the Django representation of the html forms that are used in the website. It is for instance, the login form, some administrative forms...

  - models.py: all Danjgo objects representing the tables in the database. This is an important file that describes the entire database structure.
  - scripts.py: some useful scripts, mainly used for management purposes.
  - urls.py: the URL pattern for the pages available through this app.
  - views.py: the most important file. The views are functions that are called to render the page requested by the user. This is where all the computations that are done at runtime are described.
- manage.py: Python file to run local functions

1.2 Database Structure

In the following, we provide more details about the structure of the database behind Preflib.

Let us first go through all the tables present in the database.

DATAFILE The most fundamental entity for Preflib is the datafile. The DATAFILE table contains a reference to all the datafiles that are in Preflib. The table does not contain the data in itself—it is stored in a file and not in the database—but all the relevant information about it: some basic details and datapatch in which the file is.

DATAPATCH The datapatch is the first level of classification of the datafile. It contains several datafile of different datatype. All the datafiles are based on the same preferences but the representation, the datatype, is different.

DATASET A dataset is a collection of datapatches. The datapatches will typically represent different years of the same election.

DATAPROPERTY A dataproperty is an additional information about a given datafile. It can have to do about the general properties of the data (number of candidates...) or about some more specific structure of the data (single-peakedness...).

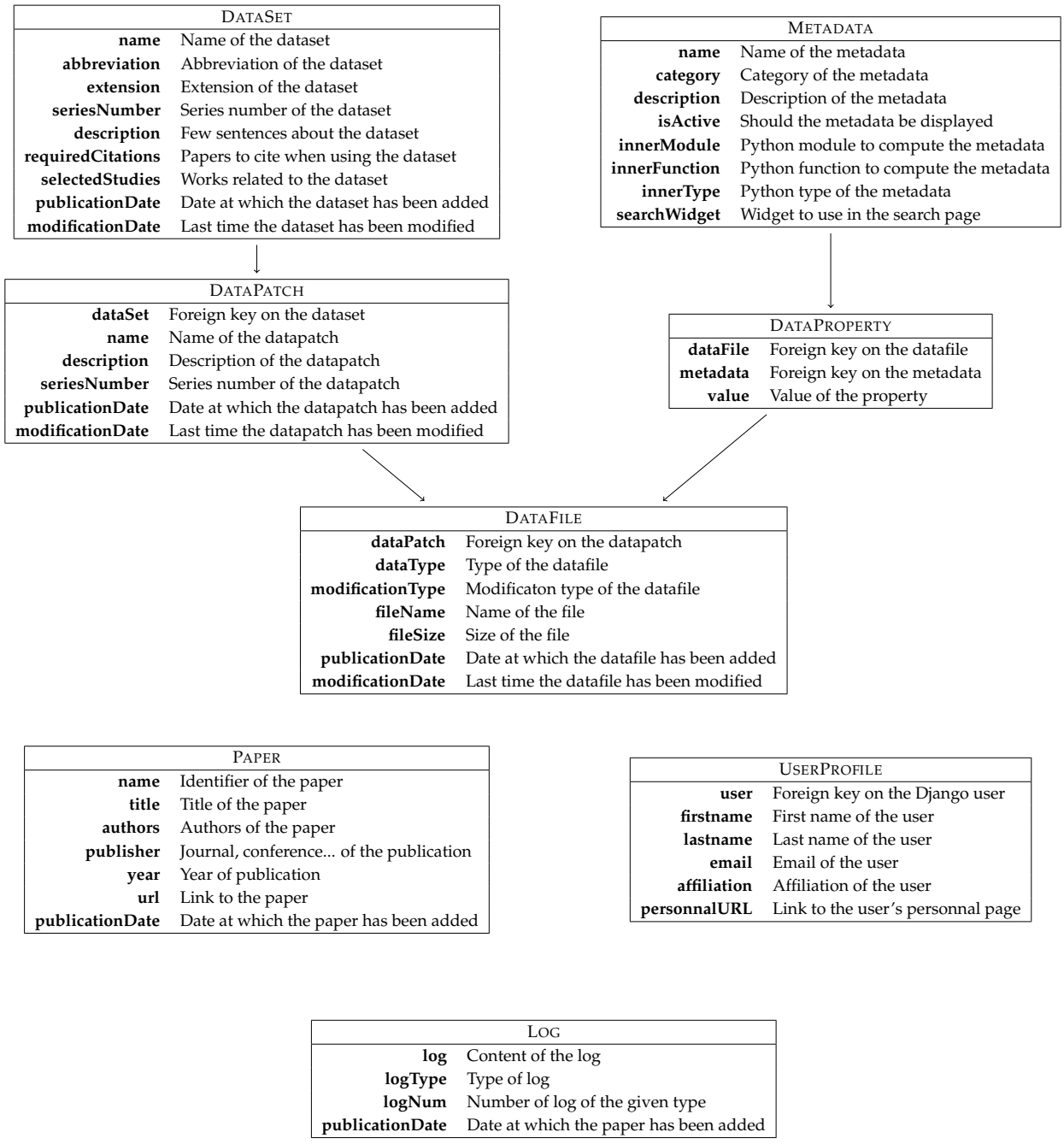
METADATA The METADATA table stores all the different metadata available in the system. Their values are in the DATAPROPERTY table.

PAPER This table stores the information about the papers which are using Preflib.

USERPROFILE All the informations about the users that are not in the Django User class are present in this table.

LOG Logs of what is happening in the inside are gathers in this table.

The following figure summarizes the links between the different tables and presents all the elements present in the tables.


DataSet
name Name of the dataset
abbreviation Abbreviation of the dataset
extension Extension of the dataset
seriesNumber Series number of the dataset
description Few sentences about the dataset
requiredCitations Papers to cite when using the dataset
selectedStudies Works related to the dataset
publicationDate Date at which the dataset has been added
modificationDate Last time the dataset has been modified
Metadata
name Name of the metadata
category Category of the metadata
description Description of the metadata
isActive Should the metadata be displayed
innerModule Python module to compute the metadata
innerFunction Python function to compute the metadata
innerType Python type of the metadata
searchWidget Widget to use in the search page
DataPatch
dataSet Foreign key on the dataset
name Name of the datapatch
description Description of the datapatch
seriesNumber Series number of the datapatch
publicationDate Date at which the datapatch has been added
modificationDate Last time the datapatch has been modified
DataProperty
dataFile Foreign key on the datafile
metadata Foreign key on the metadata
value Value of the property
DataFile
dataPatch Foreign key on the datapatch
dataType Type of the datafile
modificationType Modificaton type of the datafile
fileName Name of the file
fileSize Size of the file
publicationDate Date at which the datafile has been added
modificationDate Last time the datafile has been modified
Paper
name Identifier of the paper
title Title of the paper
authors Authors of the paper
publisher Journal, conference... of the publication
year Year of publication
url Link to the paper
publicationDate Date at which the paper has been added
UserProfile
user Foreign key on the Django user
firstname First name of the user
lastname Last name of the user
email Email of the user
affiliation Affiliation of the user
personnalURL Link to the user's personnal page
Log
log Content of the log
logType Type of log
logNum Number of log of the given type
publicationDate Date at which the paper has been added

Chapter 2

Maintaining PrefLib