billy

Release 1.8.4

November 18, 2015

Contents

billy is a suite of tools developed as a part of Open States that provide a framework for scraping, storing, and sharing legislative information.

Features:

- Scraper architecture for scraping bills, votes, legislators, committees, and events.
- Utility scripts for data cleanup and analysis.
- Bulk data export.
- Server that includes a data browser and API.

CHAPTER 1

Documentation

1.1 Writing Scrapers

A state scraper is implementing by providing classes derived from *BillScraper*, *LegislatorScraper*, *VoteScraper*, and *CommitteeScraper*.

Derived scraper classes should override the `scrape()` method that that is responsible for creating *Bill*, *Legislator*, *Vote*, and *Committee* objects as appropriate.

Example state scraper directory structure:

```
./ex/__init__.py      # metadata for "ex" state scraper
./ex/bills.py         # contains EXBillScraper (also scrapes Votes)
./ex/legislators.py   # contains EXLegislatorScraper
./ex/committees.py    # contains EXCommitteeScraper
```

1.1.1 billy.scrape

Scraper

The most useful on the base *Scraper* class is `urlopen(url, method='GET', body=None)`. `Scraper.urlopen` opens a URL and returns a string-like object that can then be parsed by a library like lxml.

This method provides advantages over built-in urlopen methods in that the underlying *Scraper* class can be configured to support rate-limiting, caching, and provides robust error handling.

Note: For advanced usage see scrapelib which provides the basis for *billy.scrape.Scraper*.

1.1.2 Logging

The base class also configures a python logger instance and provides several shortcuts for logging at various log levels:

`log(msg, *args, **kwargs)` log a message with level `logging.INFO`

`debug(msg, *args, **kwargs)` log a message with level `logging.DEBUG`

`warning(msg, *args, **kwargs)` log a message with level `logging.WARNING`

Note: It is also possible to access the `self.logger` object directly.

class billy.scrape.**Scraper**(*metadata*, *output_dir=None*, *strict_validation=None*, *fastmode=False*)
Base class for all Scrapers

Provides several useful methods for retrieving URLs and checking arguments against metadata.

__init__(*metadata*, *output_dir=None*, *strict_validation=None*, *fastmode=False*)
Create a new Scraper instance.

Parameters

- **metadata** – metadata for this scraper
- **output_dir** – the data directory to use
- **strict_validation** – exit immediately if validation fails

validate_session(*session*, *latest_only=False*)
Check that a session is present in the metadata dictionary.

raises *NoDataForPeriod* if session is invalid

Parameters **session** – string representing session to check

validate_term(*term*, *latest_only=False*)
Check that a term is present in the metadata dictionary.

raises *NoDataForPeriod* if term is invalid

Parameters

- **term** – string representing term to check
- **latest_only** – if True, will raise exception if term is not the current term (default: False)

SourcedObject

class billy.scrape.**SourcedObject**(*_type*, ***kwargs*)
Base object used for data storage.

Base class for *Bill*, *Legislator*, *Vote*, and *Committee*.

SourcedObjects work like a dictionary. It is possible to add extra data beyond the required fields by assigning to the *SourcedObject* instance like a dictionary.

add_source(*url*, ***kwargs*)
Add a source URL from which data related to this object was scraped.

Parameters **url** – the location of the source

Exceptions

class billy.scrape.**ScrapeError**(*msg*, *orig_exception=None*)
Base class for scrape errors.

class billy.scrape.**NoDataForPeriod**(*period*)
Exception to be raised when no data exists for a given period

1.1.3 Bills

BillScraper

`BillScraper` implementations should gather and save *`Bill`* objects.

Sometimes it is easiest to also gather *`Vote`* objects in a BillScraper as well, these can be attached to *`Bill`* objects via the `add_vote()` method.

class `billy.scrape.bills.`**`BillScraper`**(*metadata*, *output_dir=None*, *strict_validation=None*, *fastmode=False*)

`scrape`(*chamber*, *session*)
Grab all the bills for a given chamber and session. Must be overridden by subclasses.

Should raise a `NoDataForPeriod` exception if it is not possible to scrape bills for the given session.

Bill

class `billy.scrape.bills.`**`Bill`**(*session*, *chamber*, *bill_id*, *title*, ***kwargs*)
Object representing a piece of legislation.

See *`SourcedObject`* for notes on extra attributes/fields.

`__init__`(*session*, *chamber*, *bill_id*, *title*, ***kwargs*)
Create a new *`Bill`*.

Parameters

- **`session`** – the session in which the bill was introduced.
- **`chamber`** – the chamber in which the bill was introduced: either 'upper' or 'lower'
- **`bill_id`** – an identifier assigned to this bill by the legislature (should be unique within the context of this chamber/session) e.g.: 'HB 1', 'S. 102', 'H.R. 18'
- **`title`** – a title or short description of this bill provided by the official source

Any additional keyword arguments will be associated with this bill and stored in the database.

`add_action`(*actor*, *action*, *date*, *type=None*, *committees=None*, *legislators=None*, ***kwargs*)
Add an action that was performed on this bill.

Parameters

- **`actor`** – a string representing who performed the action. If the action is associated with one of the chambers this should be 'upper' or 'lower'. Alternatively, this could be the name of a committee, a specific legislator, or an outside actor such as 'Governor'.
- **`action`** – a string representing the action performed, e.g. 'Introduced', 'Signed by the Governor', 'Amended'
- **`date`** – the date/time this action was performed.
- **`type`** – a type classification for this action

;param committees: a committee or list of committees to associate with this action

`add_document`(*name*, *url*, *mimetype=None*, ***kwargs*)
Add a document or media item that is related to the bill. Use this method to add documents such as Fiscal Notes, Analyses, Amendments, or public hearing recordings.

Parameters

- **name** – a name given to the document, e.g. 'Fiscal Note for Amendment LCO 6544'
- **url** – link to location of document or file
- **mimetype** – MIME type of the document

If multiple formats of a document are provided, a good rule of thumb is to prefer text, followed by html, followed by pdf/word/etc.

add_source(*url*, ***kwargs*)
Add a source URL from which data related to this object was scraped.

Parameters **url** – the location of the source

add_sponsor(*type*, *name*, ***kwargs*)
Associate a sponsor with this bill.

Parameters

- **type** – the type of sponsorship, e.g. 'primary', 'cosponsor'
- **name** – the name of the sponsor as provided by the official source

add_title(*title*)
Associate an alternate title with this bill.

add_version(*name*, *url*, *mimetype=None*, *on_duplicate='error'*, ***kwargs*)
Add a version of the text of this bill.

Parameters

- **name** – a name given to this version of the text, e.g. 'As Introduced', 'Version 2', 'As amended', 'Enrolled'
- **url** – the location of this version on the legislative website.
- **mimetype** – MIME type of the document
- **on_duplicate** – What to do if a duplicate is seen: error - default option, raises a ValueError ignore - add the document twice (rarely the right choice) use_new - use the new name, removing the old document use_old - use the old name, not adding the new document

If multiple formats are provided, a good rule of thumb is to prefer text, followed by html, followed by pdf/word/etc.

add_vote(*vote*)
Associate a *Vote* object with this bill.

1.1.4 Votes

VoteScraper

`VoteScraper` implementations should gather and save *Vote* objects.

If a state's `BillScraper` gathers votes it is not necessary to provide a `VoteScraper` implementation.

class billy.scrape.votes.**VoteScraper**(*metadata*, *output_dir=None*, *strict_validation=None*, *fastmode=False*)

scrape(*chamber*, *session*)
Grab all votes for a given chamber and session. Must be overridden by subclasses.

Should raise a `NoDataForPeriod` exception if it is not possible to scrape votes for the provided session.

Vote

class `billy.scrape.votes.`**Vote**(*chamber*, *date*, *motion*, *passed*, *yes_count*, *no_count*, *other_count*, *type='other'*, ***kwargs*)

__init__(*chamber*, *date*, *motion*, *passed*, *yes_count*, *no_count*, *other_count*, *type='other'*, ***kwargs*)
Create a new *Vote*.

Parameters

- **chamber** – the chamber in which the vote was taken, 'upper' or 'lower'
- **date** – the date/time when the vote was taken
- **motion** – a string representing the motion that was being voted on
- **passed** – did the vote pass, True or False
- **yes_count** – the number of 'yes' votes
- **no_count** – the number of 'no' votes
- **other_count** – the number of abstentions, 'present' votes, or anything else not covered by 'yes' or 'no'.
- **type** – vote type classification

Any additional keyword arguments will be associated with this vote and stored in the database.

Examples:

```
Vote('upper', '', '12/7/08', 'Final passage',
     True, 30, 8, 3)
Vote('lower', 'Finance Committee', '3/4/03 03:40:22',
     'Recommend passage', 12, 1, 0)
```

add_source(*url*, ***kwargs*)
Add a source URL from which data related to this object was scraped.

Parameters **url** – the location of the source

no(*legislator*)
Indicate that a legislator (given as a string of their name) voted 'no'.

other(*legislator*)
Indicate that a legislator (given as a string of their name) abstained, voted 'present', or made any other vote not covered by 'yes' or 'no'.

yes(*legislator*)
Indicate that a legislator (given as a string of their name) voted 'yes'.

Examples:

```
vote.yes('Smith')
vote.yes('Alan Hoerth')
```

1.1.5 Legislators

`LegislatorScraper` implementations should gather and save *Legislator* objects.

Sometimes it is easiest to also gather committee memberships at the same time as legislators. Committee memberships can be attached to *Legislator* objects via the `add_role()` method.

LegislatorScraper

class billy.scrape.legislators.**LegislatorScraper**(*metadata*, *output_dir=None*, *strict_validation=None*, *fastmode=False*)

scrape(*chamber*, *term*)
Grab all the legislators who served in a given term. Must be overridden by subclasses.

Should raise a `NoDataForPeriod` exception if the year is invalid.

Person

class billy.scrape.legislators.**Person**(*full_name*, *first_name=''*, *last_name=''*, *middle_name=''*, ***kwargs*)

__init__(*full_name*, *first_name=''*, *last_name=''*, *middle_name=''*, ***kwargs*)
Create a Person.

Note: the *Legislator* class should be used when dealing with legislators.

Parameters

- **full_name** – the person's full name
- **first_name** – the first name of this legislator (if specified)
- **last_name** – the last name of this legislator (if specified)
- **middle_name** – a middle name or initial of this legislator (if specified)

add_role(*role*, *term*, *start_date=None*, *end_date=None*, ***kwargs*)
Examples:

leg.add_role('member', term='2009', chamber='upper', party='Republican', district='10th')

Legislator

class billy.scrape.legislators.**Legislator**(*term*, *chamber*, *district*, *full_name*, *first_name=''*, *last_name=''*, *middle_name=''*, *party=''*, ***kwargs*)

__init__(*term*, *chamber*, *district*, *full_name*, *first_name=''*, *last_name=''*, *middle_name=''*, *party=''*, ***kwargs*)
Create a Legislator.

Parameters

- **term** – the term for this legislator
- **chamber** – the chamber in which this legislator served, 'upper' or 'lower'

- **district** – the district this legislator is representing, as given e.g. 'District 2', '7th', 'District C'.
- **full_name** – the full name of this legislator
- **first_name** – the first name of this legislator (if specified)
- **last_name** – the last name of this legislator (if specified)
- **middle_name** – a middle name or initial of this legislator (if specified)
- **party** – the party this legislator belongs to (if specified)

Note: please only provide the first_name, middle_name and last_name parameters if they are listed on the official web site; do not try to split the legislator's full name into components yourself.

---

add_source(*url*, ***kwargs*)
Add a source URL from which data related to this object was scraped.

Parameters **url** – the location of the source

1.1.6 Committees

`CommitteeScraper` implementations should gather and save *`Committee`* objects.

If a state's `LegislatorScraper` gathers committee memberships it is not necessary to provide a `CommitteeScraper` implementation.

CommitteeScraper

class billy.scrape.committees.**CommitteeScraper**(*metadata*, *output_dir=None*, *strict_validation=None*, *fastmode=False*)

Committee

class billy.scrape.committees.**Committee**(*chamber*, *committee*, *subcommittee=None*, ***kwargs*)

__init__(*chamber*, *committee*, *subcommittee=None*, ***kwargs*)
Create a Committee.

Parameters

- **chamber** – the chamber this committee is associated with ('upper', 'lower', or 'joint')
- **committee** – the name of the committee
- **subcommittee** – the name of the subcommittee (optional)

add_member(*legislator*, *role='member'*, ***kwargs*)
Add a member to the committee object.

Parameters

- **legislator** – name of the legislator
- **role** – role that legislator holds in the committee (eg. chairman) default: 'member'

1.1.7 Events

`EventScraper` implementations should gather and save *`Event`* objects.

Relevant bills, documents, and participants can be attached to *`Event`* objects via the `add_related_bill()`, `add_document()`, and `add_participant()` methods, respectively.

EventScraper

class `billy.scrape.events.`**`EventScraper`**(*metadata*, *output_dir=None*, *strict_validation=None*, *fastmode=False*)

Event

class `billy.scrape.events.`**`Event`**(*session*, *when*, *type*, *description*, *location*, *end=None*, ***kwargs*)

1.2 State Metadata

The top level directory for a scraper should include an `__init__.py` that has a `metadata` dictionary.

This file should include the basic details on the state, most importantly the terms and sessions for which the scraper can be run for.

1.2.1 Metadata Fields

`name` The name of the state

`abbreviation` The two-letter abbreviation of the state

`legislature_name` The name of the state legislature

`upper_chamber_name` The name of the 'upper' chamber of the state legislature (if applicable)

`lower_chamber_name` The name of the 'lower' chamber of the state legislature (if applicable)

`upper_chamber_term` The length, in years, of a term for members of the 'upper' chamber (if applicable)

`lower_chamber_term` The length, in years, of a term for members of the 'lower' chamber (if applicable)

`upper_chamber_title` The title used to refer to members of the 'upper' chamber (if applicable)

`lower_chamber_title` The title used to refer to members of the 'lower' chamber (if applicable)

`latest_json_url` URL pointing to a download of all data for the state

`latest_json_date` datestamp of the file at `latest_json_url`

`latest_csv_url` URL pointing to experimental CSV download of state data

`latest_csv_date` datestamp of the file at `latest_csv_url`

`terms` A list of terms that we have data available for. Each session will be an object with the following fields:

- `start_year`: The year in which this session began.
- `end_year`: The year in which this session ended.
- `name`: The name of this session.

- `sessions`: List of sessions that took place inside the given term.

`session_details` Details about sessions.

Dictionary keys correspond to `sessions` and values are dictionaries of extra metadata about a session.

Currently the only required field is `display_name`, though fields that may be present include `start_date` and `end_date`, as well as `type` indicating whether the session was a normally scheduled or special session.

Example

Example data for Maryland:

```
import datetime

metadata = dict(
    name='Maryland',
    abbreviation='md',
    legislature_name='Maryland General Assembly',
    upper_chamber_name='Senate',
    lower_chamber_name='House of Delegates',
    upper_chamber_title='Senator',
    lower_chamber_title='Delegate',
    upper_chamber_term=4,
    lower_chamber_term=4,
    terms=[
        {'name': '2007-2010', 'sessions': ['2007', '2007s1', '2008',
                                           '2009', '2010'],
         'start_year': 2007, 'end_year': 2010},
        {'name': '2011-2014', 'sessions': ['2011'],
         'start_year': 2011, 'end_year': 2014},
    ],
    session_details={
        '2007': {'start_date': datetime.date(2007,1,10),
                 'end_date': datetime.date(2007,4,10),
                 'number': 423,
                 'type': 'primary'},
        '2007s1': {'start_date': datetime.date(2007,10,29),
                   'end_date': datetime.date(2007,11,19),
                   'number': 424,
                   'type': 'special'},
        '2008': {'start_date': datetime.date(2008,1,9),
                 'end_date': datetime.date(2008,4,7),
                 'number': 425,
                 'type': 'primary'},
        '2009': {'start_date': datetime.date(2009,1,14),
                 'end_date': datetime.date(2009,4,13),
                 'number': 426,
                 'type': 'primary'},
        '2010': {'start_date': datetime.date(2010,1,13),
                 'end_date': datetime.date(2010,4,12),
                 'number': 427,
                 'type': 'primary'},
        '2011': {'start_date': datetime.date(2011,1,12),
                 'end_date': datetime.date(2011,4,12),
                 'number': 428,
                 'type': 'primary'},
    },
)
```

1.3 Billy Configuration

Billy has a global configuration object at `billy.conf.settings` that is used in scraping, import, and serving the API.

All billy scripts attempt to load a `billy_settings` module which should either be on the import path or in the working directory, this file can contain overrides and custom settings. As of 0.9.2 if no `billy_settings` module can be located a warning message will be printed to the console.

1.3.1 Default Settings

MONGO_HOST Host or IP address of MongoDB server. (default: "localhost")

MONGO_PORT Port for MongoDB server. (default: "27017")

MONGO_DATABASE MongoDB database name. (default: "billy")

API_BASE_URL Public URL that the API can be accessed at.

SCRAPER_PATHS Paths that scraper modules are stored under, will be added to `sys.path` when attempting to load scrapers.

BILLY_DATA_DIR Directory where scraped data should be stored. (default: "<cwd>/data")

BILLY_CACHE_DIR Directory where scraper cache should be stored. (default: "<cwd>/cache")

BILLY_ERROR_DIR Directory where scraper error dumps should be stored. (default: "<cwd>/errors")

BILLY_MANUAL_DATA_DIR Directory where manual data files for matched ids/subjects are stored. (default: "<cwd>/manual_data")

BILLY_SUBJECTS List of valid subject names

SCRAPELIB_TIMEOUT Value (in seconds) for url retrieval timeout. (default: 600)

SCRAPELIB_RETRY_ATTEMPTS Number of retries to make if an unexpected failure occurs when downloading a URL. (default: 3)

SCRAPELIB_RETRY_WAIT_SECONDS Number of seconds to wait between initial attempt and first retry. (default: 20)

1.3.2 Command-Line Overrides

Most available scripts can override the above default settings with command line switches:

--mongo_host <mongo_host>
Override MONGO_HOST

--mongo_port <mongo_port>
Override MONGO_PORT

--mongo_db <mongo_db>
Override MONGO_DATABASE

-d <data_dir>, **--data_dir** <data_dir>
Override BILLY_DATA_DIR

--cache_dir <cache_dir>
Override BILLY_CACHE_DIR

--error_dir <error_dir>
Override BILLY_ERROR_DIR

--manual_data_dir <manual_data_dir>
Override BILLY_MANUAL_DATA_DIR

--retries <retries>
Override SCRAPELIB_RETRY_ATTEMPTS

--retry_wait <retry_wait>
Override SCRAPELIB_RETRY_WAIT_SECONDS

1.4 Billy Scripts

1.4.1 Scraping

billy-update <STATE>

MODULE
state scraper module name (eg. nc) [required]

--scrape, --import, --report
Which portions of the update process to run, specifying none implies --scrape --import --report

–scrape crawls the specified sites and writes data to disk in JSON format –import imports the JSON format on disk to billy's MongoDB database –report runs a series of reports on data quality and aggregates

--bills, --legislators, --votes, --committees, --events
include (bill, legislator, vote, committee, event) data in (scrape/import/report) Specifying none is the same as specifying –bills –legislators –votes –committees

--upper, --lower
process upper/lower chamber (if neither is specified will include both)

-s SESSION, **--session** SESSION
session(s) to scrape, must be present in state's metadata

-t TERM, **--term** TERM
term(s) to scrape, must be present in state's metadata

--strict
fail immediately when encountering validation warnings

-n, --no_cache
do not use cache

--fastmode
operate in "fast mode", using cached version when possible and removing –rpm induced delays

-r RPM, **--rpm** RPM
set maximum number of requests per minute (default: 60)

--timeout TIMEOUT
set HTTP timeout in seconds (default: 10s)

1.5 billy changelog

1.5.1 1.8.4

18 November 2015

- added ability to override _scraped_name field for legislators

1.5.2 1.8.3

2 November 2015

- removed ability to edit legislator offices in admin interface

1.5.3 1.8.2

29 October 2015

- enable editing of legislator offices in admin interface

1.5.4 1.8.1

18 September 2015

- add Docker deployment
- upgrade to Django 1.6
- upgrade to scrapelib 1.0
- enhance legislative division lookup
- improve CSV handling
- fix Piston errors
- pin older module versions

1.5.5 1.8.0

16 January 2014

- more CSRF bugfixing
- ensure_indexes command
- beginnings of news entry api
- add newsapi methods
- add last_action_since API parameter
- bugfix for committee having itself as parent
- bugfix for committee display on site
- bugfix for ElasticSearch parse errors causing 500s
- miscellaneous schema updates

- switch to scrapelib >= 0.8
- switch from pyes to pyelasticsearch
- simplify module loading mechanism

1.5.6 1.7.0

16 February 2013

- major changes in how ElasticSearch component works
- functional (Scout-synced) favorites support
- drastically improved mongodb indexing on bills collection
- add LOCKSMITH_REGISTRATION_URL and ENABLE_ELASTICSEARCH_PUSH settings
- add legislature_url to metadata
- more flexible support for non-2 letter abbreviations
- more robust public views for events
- allow joint committee votes
- add –pdb and related options to billy-update
- Michigan-specific exemption in fix_bill_id
- bugfix: correct centroid in boundary api method
- bugfix: stop activating legislators w/o active role
- bugfix: handle unicode in searches
- bugfix: ensure_csrf on favorites-wielding views

1.5.7 1.6.0

5 December 2012

- completion of move from 'state' to flexible LEVEL_FIELD
- new type: speeches
- documentation improvements
- beginnings of generic templates for billy.web.public
- change to metadata handling of chambers
- use updated represent-boundaries instead of django-boundaryservice
- bugfix for latest pymongo
- basic API tests

1.5.8 1.5.0

1 November 2012

- improved committee_id matching

- added bounding box to district polygon API
- added 'other_parties' to legislator schema
- events: ical support
- merging of admin view & public view a bit
- introduction of billy.core for settings & dbs
- improved action categorization in billy.scrape.actions
- bring fulltext processing in to billy
- logging colors!
- lots of cleanup & deduplication of code
- test improvements w/ fixtures now

1.5.9 1.4.0

31 August 2012

- new summary field on bills
- enable editing legislators in admin
- leg_id view replacing more manual_data csvs
- automatically attempt to link actions to votes and bills
- support fields API parameter in more places
- popularity tracking added
- fix to how roles are shown for old legislators
- **limits to number of items displayed in public view when counts are** extremely high
- basic user-account & dev-mode support
- deeper influence explorer integration
- addition of import filters
- ability to create data quality exceptions
- more tests for models

1.5.10 1.3.0

30 July 2012

- first truly usable version of billy.web.public
- remove retire, load_legislators, and prune_committees commands in favor of admin
- more admin improvements including subject support and cleaned up reporting
- new offices support on legislators
- refactor of billy.models
- db: denormalize votes into own collection on bill import
- db: add action_dates to bills

- unification of numerous settings into API_KEY
- bugfix for unicode data in dumpjson
- bugfix for name matching being too loose from manual_data
- bugfix for billy-update deleting metadata without –scrape

1.5.11 1.2.0

29 May 2012

- further development of the public site
- use elasticsearch for bill search
- improvements to event support
- **refresh of settings**
  - ENABLE_OYSTER setting replaces –oyster
  - support for module-specific settings overrides
- support for a new scrape signature (chambers vs. chamber)
- utility function for pulling data from .doc files
- bugfix for pymongo 2.2

1.5.12 1.1.0

23 April 2012

- large refactor of billy.site.{browse,www} into billy.web.{admin,public}
- require new scrapelib >= 0.7
- overhaul of event support, greatly improved schema
- scrape: improved vote validation
- API: expose internal id on all objects, including bills
- API: new method for direct lookup of bills by id
- API: added created_at sort to bills
- add support for text extraction from bills

1.5.13 1.0.0

2 April 2012

- **lots of improvements to billy admin**
  - general style overhaul
  - duplicate_versions view
- **API:**
  - removal of XML

    - removal of RSS emitter and broken stats endpoint
- **billy-update command line radically changed**
    - defaults to actually doing work
    - -vvv dropped
    - –strict dropped, –nostrict now exists
    - simplification of how –session/–term work
- drop billy-util districtcsv in favor of an admin view
- previously internal bill ids are now 8 digits
- addition of billy-update –oyster argument, adds tracking of versions
- duplicate_versions is now just that, not versions+documents
- bugfix: stop silently swallowing errors in subject csvs

1.5.14 0.9.6

27 February 2012

- add alternate_bill_ids and related functionality (needed for TN)
- updated oysterize command to work with oyster >= 0.3
- added initial work on class-based models
- added new beginning of web frontend
- added run logging work
- bugfix: billy-util broken by jenkins command
- bugfix: random_bill restricted session

1.5.15 0.9.5

21 February 2012

- added doc_ids on versions and documents
- API: add boundary_id to legislator responses (experimental)
- browse: MOM legislator merge tool
- browse: improved browse templates & random_bill
- scrapers: –cache_dir argument added
- scrapers: _partial_vote_bill_id flag added for Rhode Island
- bugfix: boundary API method returning first polygon
- bugfix: dotted keys in reports
- bugfix: billy-util retire
- bugfix: unicode error in loadlegislators

1.5.16 0.9.4

20 January 2012

- **lots of fixes and improvements to browse**
  - new /bills/ view
  - row highlighting
  - unmatched_leg_ids page
  - other_actions page
  - json views
  - random_bill/?bad_vote_counts
- **new and fixed utils**
  - districtcsv for generating district CSV stubs
  - prunecommittees for removing old committees
  - load_legislators fixed
- **improve session handling**
  - session_list in metadata file
  - missing sessions trigger an error
- new capitol_maps feature in metadata
- latest_only can be a flag on scrapers that only work for latest term
- addition of optional mimetype on documents & versions
- promote legislator's url to a non + field
- replace all csv usage with unicodecsv
- API: block requests for over 5000 bills at once

1.5.17 0.9.3

30 November 2011

- force tests to use a test database
- –mongo_host, –mongo_db, –mongo_port command line options
- sneaky_update_filter option added, can ignore minor updates
- API bugfix when chamber isn't specified on bill lookup
- change importers to use logger instead of unbuffered print statements
- **billy-update**
  - billy-scrape deprecated and replaced with billy-update
  - billy-import, billy-bill-scrape, billy-import-districts replaced
- **billy-util**
  - takes place of all utility scripts that didn't get merged into billy-update

- **reporting**
  - removed billy-generate-stats and replaced with robust reporting
  - updated browse interface to use reports
  - browse interface also got a partial facelift (more to come)

1.5.18 0.9.2

26 September 2011

- documentation improvements/moved to readthedocs.org
- load settings from a `billy_settings.py` file
- addition of `SCRAPER_PATHS` argument

1.5.19 0.9.1

23 September 2011

- packaging bugfix

1.5.20 0.9.0

23 September 2011

- initial release as used by Open States

b

Symbols

A

B

C

E

L

M

N

O

P

S

V

Y