Open States API Documentation

Release 1.0

Open States

May 18, 2017

Contents:

Documentation for Open States, its API, and how to contribute.

CHAPTER 1

Open States API

Open States provides a JSON API for accessing state legislative information.

Basics

- All API calls are URLs in the form `https://openstates.org/api/v1/METHOD/`
- Responses are JSON unless otherwise specified.
- If an error occurs the response will be a plain text error message with an appropriate HTTP error code (404 if object is not found, 401 if authentication fails, etc.).
- To use the API you must register for an API key.
- Once activated, pass your API key via the `apikey` query paramter or the `X-API-KEY` header.
- All changes to the API will be announced on the Open States Google Group. It is recommended you subscribe if you're using the API.
- For Python users, there's an official pyopenstates package available.

Data Types

Open States provides data about six core data types.

State Metadata Details on what data is available, including terms, sessions, and state-specific names for things.

Bills Details on bills & resolutions, including actions & votes.

Legislators Details on legislators, including contact details.

Committees Details on committees as they currently stand.

Events Details on upcoming events such as committee meetings and hearings.

Districts Details on districts and their boundaries.

State Metadata

Metadata Overview Get list of all states with data available and basic metadata about their status.

State Metadata Get detailed metadata for a particular state.

Metadata Fields

The following fields are available on metadata objects:

- `abbreviation` The two-letter abbreviation of the state.
- `capitol_timezone` Timezone of state capitol (e.g. 'America/New_York')
- `chambers` Dictionary mapping chamber type (upper/lower) to an object with the following fields:
  - `name` Short name of the chamber (e.g. 'House', 'Senate')
  - `title` Title of legislators in this chamber (e.g. 'Senator')
- `feature_flags` A list of which optional features are available, options include:
  - 'subjects' - bills have categorized subjects
  - 'influenceexplorer' - legislators have influence explorer ids
  - 'events' - event data is present
- `latest_csv_date` Date that the CSV file at `latest_csv_url` was generated.
- `latest_csv_url` URL from which a CSV dump of all data for this state can be obtained.
- `latest_json_date` Date that the JSON file at `latest_json_url` was generated.
- `latest_json_url` URL from which a JSON dump of all data for this state can be obtained.
- `latest_update` Last time a successful scrape was run.
- `legislature_name` Full name of legislature (e.g. 'North Carolina General Assembly')
- `legislature_url` URL to legislature's official website.
- `name` Name of state.
- `session_details` Dictionary of session names to detail dictionaries with the following keys:
  - `type` 'primary' or 'special'
  - `display_name` e.g. '2009-2010 Session'
  - `start_date` date session began
  - `end_date` date session began
- `terms` List of terms in order that they occurred. Each item in the list is comprised of the following keys:
  - `start_year` Year session started.
  - `end_year` Year session ended.
  - `name` Display name for term (e.g. '2009-2011').
  - `sessions` List of sessions (e.g. '2009'). Each session will be present in `session_details`.

Terms & Sessions

A common area for confusion, terms describe a period of time between legislative elections, for example '2009-2010'. A term can be comprised of one or more sessions:depending on how often the legislature met/adjourned within the term.

Terms are associated with legislators, while sessions are associated with bills.

Methods

Metadata Overview

This method returns just a subset (`abbreviation`, `name`, `chambers`, `feature_flags`) of metadata across all available entities.

Example: *openstates.org/api/v1/metadata/*

State Metadata

This method returns the full metadata for a state.

Example: *openstates.org/api/v1/metadata/nc/*

Examples

Metadata Overview

`openstates.org/api/v1/metadata/`


```
[
 { "name": "Alabama",
  "abbreviation": "al",
  "feature_flags": [ "subjects", "influenceexplorer" ],
  "chambers": {
   "upper": { "name": "Senate", "title": "Senator" },
   "lower": { "name": "House", "title": "Representative" }
  } },
 { "name": "Alaska",
  "abbreviation": "ak",
  "feature_flags": [ "subjects", "influenceexplorer" ],
  "chambers": {
   "upper": { "name": "Senate", "title": "Senator" },
   "lower": { "name": "House", "title": "Representative" }
  } },
 { "name": "Arizona",
  "abbreviation": "az",
  "feature_flags": [ "events", "influenceexplorer" ],
  "chambers": {
   "upper": { "name": "Senate", "title": "Senator" },
   "lower": { "name": "House", "title": "Representative" }
  } },
 { "name": "Arkansas",
  "abbreviation": "ar",
  "feature_flags": [ "influenceexplorer" ],
```

```
  "chambers": {
   "upper": { "name": "Senate", "title": "Senator" },
   "lower": { "name": "House", "title": "Representative" }
  } },
 { "name": "California",
  "abbreviation": "ca",
  "feature_flags": [ "subjects", "influenceexplorer" ],
  "chambers": {
   "upper": { "name": "Senate", "title": "Senator" },
   "lower": { "name": "Assembly", "title": "Assemblymember" }
  } },
 { "name": "Colorado",
  "abbreviation": "co",
  "feature_flags": [ "influenceexplorer" ],
  "chambers": {
   "upper": { "name": "Senate", "title": "Senator" },
   "lower": { "name": "House", "title": "Representative" }
  } },
 { "name": "Connecticut",
  "abbreviation": "ct",
  "feature_flags": [ "subjects", "events", "influenceexplorer" ],
  "chambers": {
   "upper": { "name": "Senate", "title": "Senator" },
   "lower": { "name": "House", "title": "Representative" }
  } },
 { "name": "Delaware",
  "abbreviation": "de",
  "feature_flags": [ "events", "influenceexplorer" ],
  "chambers": {
   "upper": { "name": "Senate", "title": "Senator" },
   "lower": { "name": "House", "title": "Representative" }
  } },
 { "name": "District of Columbia",
  "abbreviation": "dc",
  "feature_flags": [],
  "chambers": {
   "upper": { "name": "Council", "title": "Councilmember" }
  } },
  ...truncated...
]
```


State Metadata

`openstates.org/api/v1/metadata/nc/`


```
{
 "abbreviation": "nc",
 "capitol_timezone": "America/New_York",
 "chambers": {
  "upper": { "name": "Senate", "title": "Senator" },
  "lower": { "name": "House", "title": "Representative" }
 },
 "feature_flags": [ "subjects", "influenceexplorer" ],
 "id": "nc",
 "latest_csv_date": "2013-03-01 09:04:45",
 "latest_csv_url": "http://static.openstates.org/downloads/2013-03-01-nc-csv.zip",
```

```
 "latest_json_date": "2013-03-05 23:46:34",
 "latest_json_url": "http://static.openstates.org/downloads/2013-03-05-nc-json.zip",
 "latest_update": "2013-03-24 01:38:51",
 "legislature_name": "North Carolina General Assembly",
 "legislature_url": "http://www.ncleg.net/",
 "name": "North Carolina",
 "session_details": {
  "2009": { "type": "primary", "display_name": "2009-2010 Session", "start_date":
↪"2009-01-28 00:00:00" },
  "2011": { "type": "primary", "display_name": "2011-2012 Session", "start_date":
↪"2011-01-26 00:00:00" },
  "2013": { "type": "primary", "display_name": "2013-2014 Session", "start_date":
↪"2013-01-30 00:00:00" }
 },
 "terms": [
  { "end_year": 2010, "start_year": 2009, "name": "2009-2010", "sessions": [ "2009" ]␣
↪},
  { "end_year": 2012, "start_year": 2011, "name": "2011-2012", "sessions": [ "2011" ]␣
↪},
  { "end_year": 2014, "start_year": 2013, "name": "2013-2014", "sessions": [ "2013" ]␣
↪}
 ]
}
```


Bills

Bill Search Search bills by (almost) any of their attributes, or full text.

Bill Detail Get full detail for bill, including any actions, votes, etc.

Bill Fields

The following fields are available on bill objects:

- `state` State abbreviation.
- `session` Session key (see *State Metadata* for details).
- `bill_id` The official id of the bill (e.g. 'SB 27', 'A 2111')
- `title` The official title of the bill. Bill titles vary widely in size and content by state. Some are over 10KB long, while others are a few non-specific words, e.g. "Regarding taxes".
- `alternate_titles` List of alternate titles that the bill has had. (Often empty.)
- `action_dates` Dictionary of notable action dates (useful for determining status). Contains the following fields:
  - `first` First action (only null if there are no actions).
  - `last` Last action (only null if there are no actions).
  - `passed_lower` Date that the bill seems to have passed the lower chamber (might be null).
  - `passed_upper` Date that the bill seems to have passed the upper chamber (might be null).
  - `signed` Date that the bill appears to have signed into law (might be null).
- `actions` List of objects representing every recorded action for the bill. Action objects have the following fields:

  - `date` Date of action.
  - `action` Name of action as state provides it.
  - `actor` The chamber, person, committee, etc. responsible for this action.
  - `type` Open States-provided action categories, see *Action Types*.
- `chamber` The chamber of origination ('upper' or 'lower')
- `created_at` The date that this object first appeared in our system. (Note: not the date of introduction, see `action_dates` for that information.)
- `updated_at` The date that this object was last updated in our system. (Note: not the last action date, see `action_dates` for that information.)
- `documents` List of associated documents, see `versions` for field details.
- `id` Open States-assigned permanent ID for this bill.
- `scraped_subjects` List of subject areas that the state categorized this bill under.
- `subjects` List of Open States standardized bill subjects, see *Subjects*.
- `sources` List of source URLs used to compile information on this object.
- `sponsors` List of bill sponsors.
  - `name` Name of sponsor as it appears on state website.
  - `leg_id` Open States assigned legislator ID (will be null if no match was found).
  - `type` Type of sponsor ('primary' or 'cosponsor')
- `type` List of *Bill Types*.
- `versions` Versions of the bill text. Both documents and `versions` have the following fields:
  - `url` Official URL for this document.
  - `name` An official name for this document.
  - `mimetype` The mimetype for the document (e.g. 'text/html')
  - `doc_id` An Open States-assigned id uniquely identifying this document.
- **`votes` List of vote objects. Votes may be in the past or future;** a future vote does not have vote-outcome fields like `passed`. A vote object consists of the following fields:
  - `motion` Name of motion being voted upon (e.g. 'Passage'). The nature of these varies widely by state. Some states have a concise vocabulary, some a sloppy vocabulary. Other states include a vote ID in the motion, rendering every motion unique.
  - `chamber` Chamber vote took place in ('upper', 'lower', 'joint')
  - `date` Date of vote.
  - `passed` Boolean; true if *vote* (not bill) succeeded.
  - `id` Open States-assigned unique identifier for vote.
  - `state` State abbreviation.
  - `session` Session key (see *State Metadata* for details).
  - `sources` List of source URLs used to compile information on this object. (Can be empty if vote shares sources with bill.)
  - `yes_count` Total number of yes votes.

  - `no_count` Total number of no votes.
  - `other_count` Total number of 'other' votes (abstain, not present, etc.).
  - `yes_votes`, `no_votes`, `other_votes` List of roll calls of each type. Each is an object consisting of two keys:
    * `name` Name of voter as it appears on state website.
    * `leg_id` Open States assigned legislator ID (will be null if no match was found).

Methods

Bill Search

This method returns just a subset (`state`, `chamber`, `session`, `subjects`, `type`, `id`, `bill_id`, `title`, `created_at`, `updated_at`) of the bill fields by default.

Filter Parameters

The following parameters filter the returned set of bills, at least one must be provided.

- `state` Only return bills from a given state (e.g. 'nc')
- `chamber` Only return bills matching the provided chamber ('upper' or 'lower')
- `bill_id` Only return bills with a given bill_id.
- `bill_id__in` Accepts a pipe (|) delimited list of bill ids.
- `q` Only return bills matching the provided full text query.
- `search_window` By default all bills are searched, but if a time window is desired the following options can be passed to search_window:
  - `search_window=all` Default, include all sessions.
  - `search_window=term` Only bills from sessions within the current term.
  - `search_window=session` Only bills from the current session.
  - `search_window=session:2009` Only bills from the session named 2009.
  - `search_window=term:2009-2011` Only bills from the sessions in the 2009-2011 session.
- `updated_since` Only bills updated since a provided date (provided in YYYY-MM-DD format)
- `sponsor_id` Only bills sponsored by a given legislator id (e.g. 'ILL000555')
- `subject` Only bills categorized by Open States as belonging to this subject.
- `type` Only bills of a given type (e.g. 'bill', 'resolution', etc.)

Additional Parameters

`sort` Sort-order of results, defaults to 'last', options are:

- first
- last
- signed

- passed_lower
- passed_upper
- updated_at
- created_at

See the above `action_dates`, `created_at`, and `updated_at` documentation for the meaning of these dates.

The API will not return exceedingly large responses, so it may be necessary to use `page` and `per_page` to control the number of results returned:

- `page` Page of results, each of size `per_page` (defaults to 1)
- `per_page` Number of results per page, is unlimited unless page is set, in which case it defaults to 50.

Example: *openstates.org/api/v1/bills/?state=dc&q=taxi*

Bill Detail

This method returns the full detail object for a bill.

Example: *openstates.org/api/v1/bills/ca/20092010/AB%20667/*

Note: This method has an alternate URL form:

- `bills/openstates_bill_id` - e.g. `openstates.org/api/v1/bills/CAB00004148/` - allows lookup by bill_id

Examples

Bill Search

`openstates.org/api/v1/bills/?state=dc&q=taxi`


```
[
 {
  "title": "\"DOC INMATE PROCESSING AND RELEASE AMENDMENT ACT OF 2012\". ",
  "created_at": "2011-07-18 04:35:16",
  "updated_at": "2012-09-14 03:49:38",
  "chamber": "upper",
  "state": "dc",
  "session": "19",
  "subjects": [],
  "type": [ "bill" ],
  "id": "DCB00001021",
  "bill_id": "B 19-0428"
 },
 {
  "title": "\"TAXICAB SERVICE IMPROVEMENT AMENDMENT ACT OF 2012\".\r\n\r\n ",
  "created_at": "2012-01-06 20:53:35",
  "updated_at": "2012-12-07 20:31:54",
  "chamber": "upper",
  "state": "dc",
  "session": "19",
  "subjects": [],
  "type": [ "bill" ],
  "id": "DCB00001501",
```

```
  "bill_id": "B 19-0630"
 },
 {
  "title": "\"FISCAL YEAR 2013 BUDGET SUPPORT ACT OF 2012\". ",
  "created_at": "2012-03-27 02:19:29",
  "updated_at": "2012-10-18 03:33:02",
  "chamber": "upper",
  "state": "dc",
  "session": "19",
  "subjects": [],
  "type": [ "bill" ],
  "id": "DCB00001892",
  "bill_id": "B 19-0743"
 },
 {
  "title": "\"FISCAL YEAR 2013 BUDGET SUPPORT EMERGENCY ACT OF 2012\". ",
  "created_at": "2012-06-08 02:51:47",
  "updated_at": "2012-09-07 03:51:01",
  "chamber": "upper",
  "state": "dc",
  "session": "19",
  "subjects": [],
  "type": [ "bill" ],
  "id": "DCB00002085",
  "bill_id": "B 19-0796"
 },
 {
  "title": "\"LEON SWAIN, JR. RECOGNITION RESOLUTION OF 2012\". ",
  "created_at": "2012-04-27 02:36:38",
  "updated_at": "2012-08-22 04:20:34",
  "chamber": "upper",
  "state": "dc",
  "session": "19",
  "subjects": [],
  "type": [ "resolution" ],
  "id": "DCB00001959",
  "bill_id": "CER 19-0218"
 },
 {
  "title": "\"WASHINGTON CONVENTION CENTER ADVISORY COMMITTEE RECOGNITION RESOLUTION␣
↪OF 2011\".",
  "created_at": "2012-03-20 02:17:18",
  "updated_at": "2012-08-22 04:20:34",
  "chamber": "upper",
  "state": "dc",
  "session": "19",
  "subjects": [],
  "type": [ "resolution" ],
  "id": "DCB00001795",
  "bill_id": "CER 19-0171"
 },
 {
  "title": "\"WHEELCHAIR ACCESSIBLE TAXICABS PARITY AMENDMENT ACT OF 2011\".",
  "created_at": "2012-01-06 20:53:35",
  "updated_at": "2012-08-22 04:20:26",
  "chamber": "upper",
  "state": "dc",
  "session": "19",
```

```
  "subjects": [],
  "type": [ "bill" ],
  "id": "DCB00001506",
  "bill_id": "B 19-0635"
 },
 {
  "title": "\"FISCAL YEAR 2012 BUDGET SUPPORT ACT OF 2011\".",
  "created_at": "2011-04-06 01:53:14",
  "updated_at": "2012-10-18 03:32:58",
  "chamber": "upper",
  "state": "dc",
  "session": "19",
  "subjects": [],
  "type": [ "bill" ],
  "id": "DCB00000427",
  "bill_id": "B 19-0203"
 },
 {
  "title": "\"FISCAL YEAR 2012 BUDGET SUPPORT EMERGENCY ACT OF 2011\".\r\n ",
  "created_at": "2011-06-16 04:18:55",
  "updated_at": "2012-08-22 04:20:21",
  "chamber": "upper",
  "state": "dc",
  "session": "19",
  "subjects": [],
  "type": [ "bill" ],
  "id": "DCB00000794",
  "bill_id": "B 19-0338"
 },
 {
  "title": "\"PROFESSIONAL TAXICAB STANDARDS AND MEDALLION ESTABLISHMENT ACT OF 2011\
↪".",
  "created_at": "2011-03-21 18:55:32",
  "updated_at": "2012-08-22 04:20:17",
  "chamber": "upper",
  "state": "dc",
  "session": "19",
  "subjects": [],
  "type": [ "bill" ],
  "id": "DCB00000339",
  "bill_id": "B 19-0172"
 }
]
```


Bill Detail

`openstates.org/api/v1/bills/ca/20092010/AB%20667/`


```
{
 "action_dates": {
  "passed_upper": null,
  "passed_lower": null,
  "last": "2009-08-06 00:00:00",
  "signed": null,
  "first": "2009-02-25 00:00:00"
 },
```

```
"actions": [
 { "date": "2009-02-25 00:00:00",
  "action": "Read first time. To print.",
  "type": [ "bill:introduced", "bill:reading:1" ],
  "actor": "lower (Desk)" },
 { "date": "2009-02-26 00:00:00",
  "action": "From printer. May be heard in committee March 28.",
  "type": [ "other" ],
  "actor": "lower (Desk)" },
 { "date": "2009-03-23 00:00:00",
  "action": "Referred to Com. on HEALTH.",
  "type": [ "committee:referred" ],
  "actor": "lower (Committee CX08)" },
 { "date": "2009-04-02 00:00:00",
  "action": "From committee chair, with author's amendments: Amend, and re-refer to␣
↪Com. on HEALTH. Read second time and amended.",
  "type": [ "bill:reading:2" ],
  "actor": "lower (E&E Engrossing)" },
 { "date": "2009-04-13 00:00:00",
  "action": "Re-referred to Com. on HEALTH.",
  "type": [ "committee:referred" ],
  "actor": "lower (Committee CX08)" },
 { "date": "2009-04-15 00:00:00",
  "action": "From committee: Do pass, and re-refer to Com. on B. & P. with␣
↪recommendation: To Consent Calendar. Re-referred. (Ayes 19. Noes 0.) (April 14).",
  "type": [ "other" ],
  "actor": "lower (Committee)" },
 { "date": "2009-04-29 00:00:00",
  "action": "From committee: Do pass, and re-refer to Com. on APPR. with␣
↪recommendation: To Consent Calendar. Re-referred. (Ayes 10. Noes 0.) (April 28).",
  "type": [ "other" ],
  "actor": "lower (Committee)" },
 { "date": "2009-05-04 00:00:00",
  "action": "From committee chair, with author's amendments: Amend, and re-refer to␣
↪Com. on APPR. Read second time and amended.",
  "type": [ "bill:reading:2" ],
  "actor": "lower (E&E Engrossing)" },
 { "date": "2009-05-05 00:00:00",
  "action": "Re-referred to Com. on APPR.",
  "type": [ "committee:referred" ],
  "actor": "lower (Committee CX25)" },
 { "date": "2009-05-14 00:00:00",
  "action": "From committee: Do pass. To Consent Calendar. (May 13).",
  "type": [ "other" ],
  "actor": "lower" },
 { "date": "2009-05-18 00:00:00",
  "action": "Read second time. To Consent Calendar.",
  "type": [ "bill:reading:2" ],
  "actor": "lower" },
 { "date": "2009-05-21 00:00:00",
  "action": "Read third time, passed, and to Senate. (Ayes 77. Noes 0. Page 1628.)",
  "type": [ "other" ],
  "actor": "lower (E&E Engrossing)" },
 { "date": "2009-05-21 00:00:00",
  "action": "In Senate. Read first time. To Com. on RLS. for assignment.",
  "type": [ "bill:reading:1", "committee:referred" ],
  "actor": "upper (Rules)" },
 { "date": "2009-06-04 00:00:00",
```

```
   "action": "Referred to Com. on B., P. & E.D.",
   "type": [ "committee:referred" ],
   "actor": "upper (Committee CS42)" },
  { "date": "2009-06-22 00:00:00",
   "action": "From committee: Do pass, and re-refer to Com. on APPR. Re-referred.␣
↪(Ayes 10. Noes 0.) (June 22).",
   "type": [ "other" ],
   "actor": "upper (Committee)" },
  { "date": "2009-06-29 00:00:00",
   "action": "From committee: Be placed on second reading file pursuant to Senate␣
↪Rule 28.8.",
   "type": [ "other" ],
   "actor": "upper" },
  { "date": "2009-06-30 00:00:00",
   "action": "Read second time. To third reading.",
   "type": [ "bill:reading:2" ],
   "actor": "upper" },
  { "date": "2009-07-02 00:00:00",
   "action": "Ordered to Special Consent Calendar.",
   "type": [ "other" ],
   "actor": "upper" },
  { "date": "2009-07-09 00:00:00",
   "action": "Read third time, passed, and to Assembly. (Ayes 34. Noes 0. Page 1667.)
↪",
   "type": [ "other" ],
   "actor": "upper (Desk)" },
  { "date": "2009-07-09 00:00:00",
   "action": "In Assembly. To enrollment.",
   "type": [ "other" ],
   "actor": "lower (E&E Enrollment)" },
  { "date": "2009-07-30 00:00:00",
   "action": "Enrolled and to the Governor at 2:30 p.m.",
   "type": [ "other" ],
   "actor": "executive" },
  { "date": "2009-08-05 00:00:00",
   "action": "Approved by the Governor.",
   "type": [ "other" ],
   "actor": "executive" },
  { "date": "2009-08-06 00:00:00",
   "action": "Chaptered by Secretary of State - Chapter 119, Statutes of 2009.",
   "type": [ "other" ],
   "actor": "Secretary of State" }
 ],
 "alternate_titles": [
  "An act to amend Section 104830 of, and to add Section 104762 to, the Health and␣
↪Safety Code, relating to oral health."
 ],
 "bill_id": "AB 667",
 "chamber": "lower",
 "created_at": "2010-07-09 17:28:10",
 "documents": [],
 "id": "CAB00004148",
 "level": "state",
 "scraped_subjects": [ "Topical fluoride application." ],
 "session": "20092010",
 "sources": [
  { "url": "http://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_
↪id=200920100AB667" }
```

```
],
"sponsors": [
 { "leg_id": "CAL000044", "type": "primary", "name": "Block" }
],
"state": "ca",
"subjects": [],
"title": "An act to amend Section 1750.1 of the Business and Professions Code, and␣
↪to amend Section 104830 of, and to add Section 104762 to, the Health and Safety␣
↪Code, relating to oral health.",
"type": [ "bill", "fiscal committee" ],
"updated_at": "2012-04-06 17:17:37",
"versions": [
 {
  "url": "http://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_
↪id=200920100AB667",
  "mimetype": "text/html", "doc_id": "CAD00040031", "name": "AB667"
 }
],
"votes": [
 {
  "other_count": 6, "+threshold": "1/2",
  "other_votes": [
   { "leg_id": "CAL000014", "name": "Ashburn" },
   { "leg_id": "CAL000036", "name": "Calderon" },
   { "leg_id": "CAL000010", "name": "Corbett" },
   { "leg_id": "CAL000026", "name": "Harman" },
   { "leg_id": "CAL000021", "name": "Oropeza" },
   { "leg_id": "CAL000005", "name": "Wolk" }
  ],
  "yes_count": 34,
  "yes_votes": [
   { "leg_id": "CAL000004", "name": "Aanestad" },
   { "leg_id": "CAL000039", "name": "Alquist" },
   { "leg_id": "CAL000029", "name": "Benoit" },
   { "leg_id": "CAL000017", "name": "Cedillo" },
   { "leg_id": "CAL000011", "name": "Cogdill" },
   { "leg_id": "CAL000037", "name": "Correa" },
   { "leg_id": "CAL000001", "name": "Cox" },
   { "leg_id": "CAL000007", "name": "DeSaulnier" },
   { "leg_id": "CAL000032", "name": "Denham" },
   { "leg_id": "CAL000038", "name": "Ducheny" },
   { "leg_id": "CAL000023", "name": "Dutton" },
   { "leg_id": "CAL000033", "name": "Florez" },
   { "leg_id": "CAL000009", "name": "Hancock" },
   { "leg_id": "CAL000027", "name": "Hollingsworth" },
   { "leg_id": "CAL000022", "name": "Huff" },
   { "leg_id": "CAL000030", "name": "Kehoe" },
   { "leg_id": "CAL000003", "name": "Leno" },
   { "leg_id": "CAL000016", "name": "Liu" },
   { "leg_id": "CAL000080", "name": "Lowenthal" },
   { "leg_id": "CAL000012", "name": "Maldonado" },
   { "leg_id": null, "name": "Negrete McLeod" },
   { "leg_id": "CAL000034", "name": "Padilla" },
   { "leg_id": "CAL000018", "name": "Pavley" },
   { "leg_id": "CAL000040", "name": "Price" },
   { "leg_id": "CAL000019", "name": "Romero" },
   { "leg_id": "CAL000013", "name": "Runner" },
   { "leg_id": "CAL000031", "name": "Simitian" },
```

```
    { "leg_id": "CAL000006", "name": "Steinberg" },
    { "leg_id": "CAL000015", "name": "Strickland" },
    { "leg_id": "CAL000025", "name": "Walters" },
    { "leg_id": "CAL000002", "name": "Wiggins" },
    { "leg_id": "CAL000035", "name": "Wright" },
    { "leg_id": "CAL000028", "name": "Wyland" },
    { "leg_id": "CAL000008", "name": "Yee" }
   ],
   "no_count": 0,
   "motion": "Special Consent #12 AB667 Block By Alquist",
   "chamber": "upper",
   "state": "ca",
   "session": "20092010",
   "sources": [],
   "passed": true,
   "date": "2009-07-09 16:50:00",
   "vote_id": "CAV00009230",
   "type": "other",
   "id": "CAV00009230",
   "bill_id": "CAB00004148",
   "no_votes": []
  }
 ]
}
```


Legislators

Legislator Search Search legislators by their attributes.

Legislator Detail Get full detail for a legislator, including all roles.

Geo Lookup Lookup all legislators that serve districts containing a given point.

Legislator Fields

The following fields are available on legislator objects:

- `leg_id` Legislator's permanent Open States ID. (e.g. 'ILL000555', 'NCL000123')
- `state` Legislator's state.
- `active` Boolean value indicating whether or not the legislator is currently in office.
- `chamber` Chamber the legislator is currently serving in if active ('upper' or 'lower')
- `district` District the legislator is currently serving in if active (e.g. '7', '6A')
- `party` Party the legislator is currently representing if active.
- `email` Legislator's primary email address.
- `full_name` Full display name for legislator.
- `first_name` First name of legislator.
- `middle_name` Middle name of legislator.
- `last_name` Last name of legislator.
- `suffixes` Name suffixes (e.g. 'Jr.', 'III') of legislator.

- `photo_url` URL of an official photo of this legislator.
- `url` URL of an official webpage for this legislator.
- `created_at` The date that this object first appeared in our system.
- `updated_at` The date that this object was last updated in our system.
- `created_at` Date at which this legislator was added to our system.
- `updated_at` Date at which this legislator was last updated.
- `offices` List of office objects representing contact details for the legislator. Comprised of the following fields:
  - `type` 'capitol' or 'district'
  - `name` Name of the address (e.g. 'Council Office', 'District Office')
  - `address` Street address.
  - `phone` Phone number.
  - `fax` Fax number.
  - `email` Email address. *Any of these fields may be ''null'' if not found.*
- `roles` List of currently active *role objects* if legislator is in office.
- `old_roles` Dictionary mapping term keys to lists of roles that were valid for that term.

Roles

`roles` and `old_roles` are comprised of role objects.

Role objects can have the following fields:

- `term` Term key for this role. (See metadata *notes on terms and sessions* for details.)
- `chamber`
- `state`
- `start_date` (optional)
- `end_date` (optional)
- `type` 'member' or 'committee member'

If the role type is 'member':

- `party`
- `district`

And if the type is 'committee member':

- `committee` name of parent committee
- `subcommittee` name of subcommittee (if null, membership is just for a committee)
- `committee_id` Open States id for committee that legislator is a member of
- `position` position on committee
- `old_roles`
- `sources` List of URLs used in gathering information for this legislator.

Methods

Legislator Search

This method allows looking up a legislator by a number of parameters, the results do not include the `roles` or `old_roles` items by default.

Parameters

- `state` Filter by state.
- `first_name` Filter by first name.
- `last_name` Filter by last name.
- `chamber` Only legislators with a role in the specified chamber.
- `active` 'true' (default) to only include current legislators, 'false' will include all legislators
- `term` Only legislators that have a role in a certain term.
- `district` Only legislators that have represented the specified district.
- `party` Only legislators that have been associated with a specified party.

Example: *openstates.org/api/v1/legislators/?state=dc&chamber=upper*

Legislator Detail

This method returns the full detail for a legislator.

Example: *openstates.org/api/v1/legislators/DCL000012/*

Geo Lookup

Lookup all legislators serving districts containing a given location.

Example: *openstates.org/api/v1/legislators/geo/?lat=35.79&long=-78.78*

Examples

Legislator Search

`openstates.org/api/v1/legislators/?state=dc&chamber=upper`


```
[
 {
  "first_name": "Anita",
  "last_name": "Bonds",
  "middle_name": "",
  "district": "At-Large",
  "chamber": "upper",
  "url": "http://dccouncil.us/council/anita-bonds",
  "created_at": "2013-01-07 21:05:06",
  "updated_at": "2013-03-26 03:22:24",
```

```
 "email": "abonds@dccouncil.us",
 "active": true,
 "state": "dc",
 "offices": [
  {
   "fax": "(202) 724-8099",
   "name": "Council Office",
   "phone": "(202) 724-8064",
   "address": "1350 Pennsylvania Avenue NW, Suite 408, Washington, DC 20004",
   "type": "capitol",
   "email": null
  }
 ],
 "full_name": "Anita Bonds",
 "leg_id": "DCL000021",
 "party": "Democratic",
 "suffixes": "",
 "id": "DCL000021",
 "photo_url": "http://dccouncil.us/files/user_uploads/member_photos/AAA_small.jpg"
},
{
 "+fax": "(202) 724-8099",
 "last_name": "Mendelson",
 "updated_at": "2013-03-26 03:20:14",
 "full_name": "Phil Mendelson",
 "id": "DCL000005",
 "first_name": "Phil",
 "middle_name": "",
 "district": "Chairman",
 "office_address": "1350 Pennsylvania Avenue NW, Suite 402, Washington, DC 20004",
 "state": "dc",
 "votesmart_id": "72089",
 "party": "Democratic",
 "email": "pmendelson@dccouncil.us",
 "leg_id": "DCL000005",
 "active": true,
 "photo_url": "http://dccouncil.us/files/user_uploads/member_photos/mendelson.jpg",
 "level": "state",
 "url": "http://dccouncil.us/council/phil-mendelson",
 "created_at": "2011-02-17 22:43:55",
 "chamber": "upper",
 "offices": [
  {
   "fax": "(202) 724-8099",
   "name": "Council Office",
   "phone": "(202) 724-8032     ",
   "address": "1350 Pennsylvania Avenue NW, Suite 504, Washington, DC 20004",
   "type": "capitol",
   "email": null
  }
 ],
 "suffixes": "",
 "+phone": "(202) 724-8064      "
},
{
 "first_name": "David",
 "last_name": "Grosso",
 "middle_name": "",
```

```
  "district": "At-Large",
  "chamber": "upper",
  "url": "http://dccouncil.us/council/david-grosso",
  "created_at": "2013-01-07 21:05:06",
  "updated_at": "2013-03-26 03:22:24",
  "email": "dgrosso@dccouncil.us",
  "active": true,
  "state": "dc",
  "offices": [
   {
    "fax": "(202) 724-8071",
    "name": "Council Office",
    "phone": "(202) 724-8105",
    "address": "1350 Pennsylvania Avenue NW, Suite 406, Washington, DC 20004",
    "type": "capitol",
    "email": null
   }
  ],
  "full_name": "David Grosso",
  "leg_id": "DCL000020",
  "party": "Independent",
  "suffixes": "",
  "id": "DCL000020",
  "photo_url": "http://dccouncil.us/files/user_uploads/member_photos/david_grosso_
↪color__small.jpg"
 },
 {
  "+fax": "(202) 741-0911",
  "last_name": "Alexander",
  "updated_at": "2013-03-26 03:22:24",
  "full_name": "Yvette Alexander",
  "id": "DCL000010",
  "first_name": "Yvette",
  "middle_name": "",
  "district": "Ward 7",
  "office_address": "1350 Pennsylvania Avenue, Suite 400, NW Washington, DC 20004",
  "state": "dc",
  "votesmart_id": "72072",
  "party": "Democratic",
  "email": "yalexander@dccouncil.us",
  "leg_id": "DCL000010",
  "active": true,
  "photo_url": "http://dccouncil.us/files/user_uploads/member_photos/alexander_
↪dec2011.jpg",
  "level": "state",
  "url": "http://dccouncil.us/council/yvette-alexander",
  "created_at": "2011-02-17 22:43:55",
  "chamber": "upper",
  "offices": [
   {
    "fax": "(202) 741-0911",
    "name": "Council Office",
    "phone": "(202) 724-8068",
    "address": "1350 Pennsylvania Avenue, Suite 400, NW Washington, DC 20004",
    "type": "capitol",
    "email": null
   }
  ],
```

```
 "+phone": "(202) 724-8068",
 "suffixes": ""
},
{
 "+fax": "(202) 724-8054",
 "last_name": "Wells",
 "updated_at": "2013-03-26 03:22:24",
 "full_name": "Tommy Wells",
 "id": "DCL000008",
 "first_name": "Tommy",
 "middle_name": "",
 "district": "Ward 6",
 "office_address": "1350 Pennsylvania Avenue, Suite 408, NW Washington, DC 20004",
 "state": "dc",
 "votesmart_id": "72071",
 "party": "Democratic",
 "email": "twells@dccouncil.us",
 "leg_id": "DCL000008",
 "active": true,
 "photo_url": "http://dccouncil.us/files/user_uploads/member_photos/wells2.jpg",
 "level": "state",
 "url": "http://dccouncil.us/council/tommy-wells",
 "created_at": "2011-02-17 22:43:55",
 "chamber": "upper",
 "offices": [
  {
   "fax": "(202) 724-8054",
   "name": "Council Office",
   "phone": "(202) 724-8072",
   "address": "1350 Pennsylvania Avenue, Suite 402, NW Washington, DC 20004",
   "type": "capitol",
   "email": null
  }
 ],
 "+phone": "(202) 724-8072",
 "suffixes": ""
},
{
 "+fax": "(202) 727-8210",
 "last_name": "Orange",
 "updated_at": "2013-03-26 03:22:24",
 "full_name": "Vincent Orange",
 "id": "DCL000014",
 "first_name": "Vincent",
 "middle_name": "",
 "district": "At-Large",
 "office_address": "1350 Pennsylvania Avenue NW, Suite 107, Washington, DC 20004",
 "state": "dc",
 "party": "Democratic",
 "email": "vorange@dccouncil.us",
 "leg_id": "DCL000014",
 "active": true,
 "photo_url": "http://dccouncil.us/files/user_uploads/member_photos/orange.jpg",
 "level": "state",
 "url": "http://dccouncil.us/council/vincent-orange",
 "created_at": "2011-05-12 02:08:19",
 "chamber": "upper",
 "offices": [
```

```
    {
     "fax": "(202) 727-8210",
     "name": "Council Office",
     "phone": "(202) 724-8174      ",
     "address": "1350 Pennsylvania Avenue NW, Suite 107, Washington, DC 20004",
     "type": "capitol",
     "email": null
    }
   ],
   "+phone": "(202) 724-8174      ",
   "suffixes": ""
  },
  {
   "+fax": "(202) 741-0908",
   "last_name": "Bowser",
   "updated_at": "2013-03-26 03:22:24",
   "full_name": "Muriel Bowser",
   "id": "DCL000011",
   "first_name": "Muriel",
   "middle_name": "",
   "district": "Ward 4",
   "office_address": "1350 Pennsylvania Avenue, Suite 110, NW Washington, DC 20004",
   "state": "dc",
   "votesmart_id": "72064",
   "party": "Democratic",
   "email": "mbowser@dccouncil.us",
   "leg_id": "DCL000011",
   "active": true,
   "photo_url": "http://dccouncil.us/files/user_uploads/member_photos/Bowser_Official_
↪Photo_2012_small.jpg",
   "level": "state",
   "url": "http://dccouncil.us/council/muriel-bowser",
   "created_at": "2011-02-17 22:43:55",
   "chamber": "upper",
   "offices": [
    {
     "fax": "(202) 741-0908",
     "name": "Council Office",
     "phone": "(202) 724-8052",
     "address": "1350 Pennsylvania Avenue, Suite 110, NW Washington, DC 20004",
     "type": "capitol",
     "email": null
    }
   ],
   "suffixes": "",
   "+phone": "(202) 724-8052"
  },
  {
   "+fax": "(202) 724-8087",
   "last_name": "Catania",
   "updated_at": "2013-03-26 03:22:24",
   "full_name": "David Catania",
   "id": "DCL000003",
   "first_name": "David",
   "middle_name": "",
   "district": "At-Large",
   "office_address": "1350 Pennsylvania Avenue NW, Suite 404, Washington, DC 20004",
   "state": "dc",
```

```
  "votesmart_id": "72081",
  "party": "Independent",
  "email": "dcatania@dccouncil.us",
  "leg_id": "DCL000003",
  "active": true,
  "photo_url": "http://dccouncil.us/files/user_uploads/member_photos/catania.jpg",
  "level": "state",
  "url": "http://dccouncil.us/council/david-catania",
  "created_at": "2011-02-17 22:43:55",
  "chamber": "upper",
  "offices": [
   {
    "fax": "(202) 724-8087",
    "name": "Council Office",
    "phone": "(202) 724-7772      ",
    "address": "1350 Pennsylvania Avenue NW, Suite 404, Washington, DC 20004",
    "type": "capitol",
    "email": null
   }
  ],
  "+phone": "(202) 724-7772      ",
  "suffixes": ""
 },
 {
  "+fax": "(202) 724-8076",
  "last_name": "McDuffie",
  "updated_at": "2013-03-26 03:22:24",
  "full_name": "Kenyan McDuffie",
  "id": "DCL000017",
  "first_name": "Kenyan",
  "middle_name": "",
  "district": "Ward 5",
  "office_address": "1350 Pennsylvania Avenue NW, Suite 410, Washington, DC 20004",
  "state": "dc",
  "party": "Democratic",
  "email": "kmcduffie@dccouncil.us",
  "leg_id": "DCL000017",
  "active": true,
  "photo_url": "http://dccouncil.us/files/user_uploads/member_photos/Councilmember_
↪Kenyan_R._McDuffie_Official_Photograph_small.jpg",
  "level": "state",
  "url": "http://dccouncil.us/council/kenyan-mcduffie",
  "created_at": "2012-05-31 02:28:23",
  "chamber": "upper",
  "offices": [
   {
    "fax": "(202) 724-8076",
    "name": "Council Office",
    "phone": "(202) 724-8028 ",
    "address": "1350 Pennsylvania Avenue NW, Suite 506, Washington, DC 20004",
    "type": "capitol",
    "email": null
   }
  ],
  "suffixes": "",
  "+phone": "(202) 724-8028 "
 },
 {
```

```
  "+fax": "(202) 724-8023",
  "last_name": "Evans",
  "updated_at": "2013-03-26 03:22:24",
  "full_name": "Jack Evans",
  "id": "DCL000009",
  "first_name": "Jack",
  "middle_name": "",
  "district": "Ward 2",
  "office_address": "1350 Pennsylvania Avenue, Suite 106, NW Washington, DC 20004",
  "state": "dc",
  "votesmart_id": "72044",
  "party": "Democratic",
  "email": "jevans@dccouncil.us",
  "leg_id": "DCL000009",
  "active": true,
  "photo_url": "http://dccouncil.us/files/user_uploads/member_photos/evans.jpg",
  "level": "state",
  "url": "http://dccouncil.us/council/jack-evans",
  "created_at": "2011-02-17 22:43:55",
  "chamber": "upper",
  "offices": [
   {
    "fax": "(202) 724-8023",
    "name": "Council Office",
    "phone": "(202) 724-8058",
    "address": "1350 Pennsylvania Avenue, Suite 106, NW Washington, DC 20004",
    "type": "capitol",
    "email": null
   }
  ],
  "+phone": "(202) 724-8058",
  "suffixes": ""
 },
 {
  "+fax": "(202) 724-8109",
  "last_name": "Graham",
  "updated_at": "2013-03-26 03:22:24",
  "full_name": "Jim Graham",
  "id": "DCL000007",
  "first_name": "Jim",
  "middle_name": "",
  "district": "Ward 1",
  "office_address": "1350 Pennsylvania Avenue, Suite 105, NW Washington, DC 20004",
  "state": "dc",
  "votesmart_id": "72038",
  "party": "Democratic",
  "email": "jgraham@dccouncil.us",
  "leg_id": "DCL000007",
  "active": true,
  "photo_url": "http://dccouncil.us/files/user_uploads/member_photos/graham.jpg",
  "level": "state",
  "url": "http://dccouncil.us/council/jim-graham",
  "created_at": "2011-02-17 22:43:55",
  "chamber": "upper",
  "offices": [
   {
    "fax": "(202) 724-8109",
    "name": "Council Office",
```

```
    "phone": "(202) 724-8181",
    "address": "1350 Pennsylvania Avenue, Suite 105, NW Washington, DC 20004",
    "type": "capitol",
    "email": null
   }
  ],
  "+phone": "(202) 724-8181",
  "suffixes": ""
 },
 {
  "+fax": "(202) 724-8118",
  "last_name": "Cheh",
  "updated_at": "2013-03-26 03:22:24",
  "full_name": "Mary M Cheh",
  "id": "DCL000002",
  "first_name": "Mary",
  "middle_name": "M",
  "district": "Ward 3",
  "office_address": "1350 Pennsylvania Avenue, Suite 108, NW  Washington, DC 20004",
  "state": "dc",
  "votesmart_id": "72047",
  "party": "Democratic",
  "email": "mcheh@dccouncil.us",
  "leg_id": "DCL000002",
  "active": true,
  "photo_url": "http://dccouncil.us/files/user_uploads/member_photos/cheh.jpg",
  "level": "state",
  "url": "http://dccouncil.us/council/mary-m.-cheh",
  "created_at": "2011-02-17 22:43:55",
  "chamber": "upper",
  "offices": [
   {
    "fax": "(202) 724-8118",
    "name": "Council Office",
    "phone": "(202) 724-8062",
    "address": "1350 Pennsylvania Avenue, Suite 108, NW  Washington, DC 20004",
    "type": "capitol",
    "email": null
   }
  ],
  "+phone": "(202) 724-8062",
  "suffixes": ""
 },
 {
  "+fax": "(202) 724-8055",
  "last_name": "Barry",
  "updated_at": "2013-03-26 03:22:24",
  "full_name": "Marion Barry",
  "id": "DCL000012",
  "first_name": "Marion",
  "middle_name": "",
  "district": "Ward 8",
  "office_address": "1350 Pennsylvania Avenue NW, Suite 102, Washington, DC 20004",
  "state": "dc",
  "votesmart_id": "72074",
  "party": "Democratic",
  "email": "mbarry@dccouncil.us",
  "leg_id": "DCL000012",
```

```
  "active": true,
  "photo_url": "http://dccouncil.us/files/user_uploads/member_photos/barry.jpg",
  "level": "state",
  "url": "http://dccouncil.us/council/marion-barry",
  "created_at": "2011-02-17 22:43:55",
  "chamber": "upper",
  "offices": [
   {
    "fax": "(202) 724-8055",
    "name": "Council Office",
    "phone": "(202) 724-8045",
    "address": "1350 Pennsylvania Avenue NW, Suite 102, Washington, DC 20004",
    "type": "capitol",
    "email": null
   }
  ],
  "+phone": "(202) 724-8045",
  "suffixes": ""
 }
]
```


Legislator Detail

`openstates.org/api/v1/legislators/DCL000012/`


```
{
 "active": true,
 "chamber": "upper",
 "created_at": "2011-02-17 22:43:55",
 "district": "Ward 8",
 "email": "mbarry@dccouncil.us",
 "first_name": "Marion",
 "full_name": "Marion Barry",
 "id": "DCL000012",
 "last_name": "Barry",
 "leg_id": "DCL000012",
 "level": "state",
 "middle_name": "",
 "office_address": "1350 Pennsylvania Avenue NW, Suite 102, Washington, DC 20004",
 "offices": [
  {
   "fax": "(202) 724-8055",
   "name": "Council Office",
   "phone": "(202) 724-8045",
   "address": "1350 Pennsylvania Avenue NW, Suite 102, Washington, DC 20004",
   "type": "capitol",
   "email": null
  }
 ],
 "old_roles": {
  "2011-2012": [
   {
    "term": "2011-2012",
    "end_date": null,
    "district": "Ward 8",
    "chamber": "upper",
```

```
  "state": "dc",
  "party": "Democratic",
  "type": "member",
  "start_date": null
 },
 {
  "term": "2011-2012",
  "committee_id": "DCC000017",
  "chamber": "upper",
  "state": "dc",
  "subcommittee": null,
  "committee": "Finance and Revenue",
  "position": "member",
  "type": "committee member"
 },
 {
  "term": "2011-2012",
  "committee_id": "DCC000027",
  "chamber": "upper",
  "state": "dc",
  "subcommittee": null,
  "committee": "Jobs and Workforce Development",
  "position": "member",
  "type": "committee member"
 },
 {
  "term": "2011-2012",
  "committee_id": "DCC000021",
  "chamber": "upper",
  "state": "dc",
  "subcommittee": null,
  "committee": "the Judiciary",
  "position": "member",
  "type": "committee member"
 },
 {
  "term": "2011-2012",
  "committee_id": "DCC000019",
  "chamber": "upper",
  "state": "dc",
  "subcommittee": null,
  "committee": "Aging and Community Affairs",
  "position": "member",
  "type": "committee member"
 },
 {
  "term": "2011-2012",
  "committee_id": "DCC000026",
  "chamber": "upper",
  "state": "dc",
  "subcommittee": null,
  "committee": "Economic Development and Housing",
  "position": "member",
  "type": "committee member"
 },
 {
  "term": "2011-2012",
  "committee_id": "DCC000014",
```

```
    "chamber": "upper",
    "state": "dc",
    "subcommittee": null,
    "committee": "Human Services",
    "position": "member",
    "type": "committee member"
   },
   {
    "term": "2011-2012",
    "committee_id": "DCC000023",
    "chamber": "upper",
    "state": "dc",
    "subcommittee": null,
    "committee": "Health",
    "position": "member",
    "type": "committee member"
   }
  ]
 },
 "party": "Democratic",
 "photo_url": "http://dccouncil.us/files/user_uploads/member_photos/barry.jpg",
 "roles": [
  {
   "term": "2013-2014",
   "end_date": null,
   "district": "Ward 8",
   "chamber": "upper",
   "state": "dc",
   "party": "Democratic",
   "type": "member",
   "start_date": null
  },
  {
   "term": "2013-2014",
   "committee_id": "DCC000014",
   "chamber": "upper",
   "state": "dc",
   "subcommittee": null,
   "committee": "Human Services",
   "position": "member",
   "type": "committee member"
  },
  {
   "term": "2013-2014",
   "committee_id": "DCC000017",
   "chamber": "upper",
   "state": "dc",
   "subcommittee": null,
   "committee": "Finance and Revenue",
   "position": "member",
   "type": "committee member"
  },
  {
   "term": "2013-2014",
   "committee_id": "DCC000032",
   "chamber": "upper",
   "state": "dc",
   "subcommittee": null,
```

```
    "committee": "Education",
    "position": "member",
    "type": "committee member"
   },
   {
    "term": "2013-2014",
    "committee_id": "DCC000031",
    "chamber": "upper",
    "state": "dc",
    "subcommittee": null,
    "committee": "Workforce and Community Affairs",
    "position": "member",
    "type": "committee member"
   }
  ],
  "sources": [ { "url": "http://dccouncil.us/council/marion-barry" } ],
  "state": "dc",
  "suffixes": "",
  "updated_at": "2013-03-26 03:22:24",
  "url": "http://dccouncil.us/council/marion-barry",
  "votesmart_id": "72074"
}
```


Geo Lookup

`openstates.org/api/v1/legislators/geo/?lat=35.79&long=-78.78`


```
[
 {
  "last_name": "Stein",
  "suffix": "",
  "updated_at": "2013-03-27 02:35:39",
  "sources": [ { "url": "http://www.ncga.state.nc.us/gascripts/members/viewMember.pl?
↪sChamber=Senate&nUserID=267" } ],
  "full_name": "Josh Stein",
  "old_roles": {
   "2009-2010": [
    {
     "term": "2009-2010",
     "end_date": null,
     "district": "16",
     "level": "state",
     "chamber": "upper",
     "state": "nc",
     "party": "Democratic",
     "type": "member",
     "start_date": null
    },
    {
     "term": "2009-2010",
     "committee_id": "NCC000002",
     "level": "state",
     "chamber": "upper",
     "state": "nc",
     "subcommittee": null,
     "committee": "Appropriations on Department of Transportation",
```

```
    "type": "committee member"
   },
   {
    "term": "2009-2010",
    "committee_id": "NCC000008",
    "level": "state",
    "chamber": "upper",
    "state": "nc",
    "subcommittee": null,
    "committee": "Appropriations/Base Budget",
    "type": "committee member"
   },
   {
    "term": "2009-2010",
    "committee_id": "NCC000009",
    "level": "state",
    "chamber": "upper",
    "state": "nc",
    "subcommittee": null,
    "committee": "Commerce",
    "type": "committee member"
   },
   {
    "term": "2009-2010",
    "committee_id": "NCC000010",
    "level": "state",
    "chamber": "upper",
    "state": "nc",
    "subcommittee": null,
    "committee": "Education/Higher Education",
    "type": "committee member"
   },
   {
    "term": "2009-2010",
    "committee_id": "NCC000073",
    "level": "state",
    "chamber": "upper",
    "state": "nc",
    "subcommittee": null,
    "committee": "Finance",
    "type": "committee member"
   },
   {
    "term": "2009-2010",
    "committee_id": "NCC000012",
    "level": "state",
    "chamber": "upper",
    "state": "nc",
    "subcommittee": null,
    "committee": "Health Care",
    "type": "committee member"
   },
   {
    "term": "2009-2010",
    "committee_id": "NCC000074",
    "level": "state",
    "chamber": "upper",
    "state": "nc",
```

```
    "subcommittee": null,
    "committee": "Judiciary I",
    "type": "committee member"
   },
   {
    "term": "2009-2010",
    "committee_id": "NCC000022",
    "level": "state",
    "chamber": "upper",
    "state": "nc",
    "subcommittee": null,
    "committee": "Select Committee on Economic Recovery",
    "type": "committee member"
   },
   {
    "term": "2009-2010",
    "committee_id": "NCC000024",
    "level": "state",
    "chamber": "upper",
    "state": "nc",
    "subcommittee": null,
    "committee": "Select Committee on Energy, Science and Technology",
    "type": "committee member"
   }
  ],
  "2011-2012": [
   {
    "term": "2011-2012",
    "end_date": null,
    "district": "16",
    "chamber": "upper",
    "state": "nc",
    "party": "Democratic",
    "type": "member",
    "start_date": null
   },
   {
    "term": "2011-2012",
    "committee_id": "NCC000009",
    "chamber": "upper",
    "state": "nc",
    "subcommittee": null,
    "committee": "Commerce",
    "position": "member",
    "type": "committee member"
   },
   {
    "term": "2011-2012",
    "committee_id": "NCC000100",
    "chamber": "upper",
    "state": "nc",
    "subcommittee": null,
    "committee": "Education / Higher Education",
    "position": "member",
    "type": "committee member"
   },
   {
    "term": "2011-2012",
```

```
    "committee_id": "NCC000073",
    "chamber": "upper",
    "state": "nc",
    "subcommittee": null,
    "committee": "Finance",
    "position": "member",
    "type": "committee member"
   },
   {
    "term": "2011-2012",
    "committee_id": "NCC000074",
    "chamber": "upper",
    "state": "nc",
    "subcommittee": null,
    "committee": "Judiciary I",
    "position": "member",
    "type": "committee member"
   },
   {
    "term": "2011-2012",
    "committee_id": "NCC000018",
    "chamber": "upper",
    "state": "nc",
    "subcommittee": null,
    "committee": "Rules and Operations of the Senate",
    "position": "member",
    "type": "committee member"
   }
  ]
 },
 "id": "NCL000047",
 "first_name": "Josh",
 "middle_name": "",
 "district": "16",
 "state": "nc",
 "votesmart_id": "102971",
 "party": "Democratic",
 "email": "Josh.Stein@ncleg.net",
 "leg_id": "NCL000047",
 "boundary_id": "sldu/nc-16",
 "active": true,
 "transparencydata_id": "d3917a35b626477a9a7afaf7dbf206be",
 "photo_url": "http://www.ncga.state.nc.us/Senate/pictures/hiRes/267.jpg",
 "roles": [
  {
   "term": "2013-2014",
   "end_date": null,
   "district": "16",
   "chamber": "upper",
   "state": "nc",
   "party": "Democratic",
   "type": "member",
   "start_date": null
  },
  {
   "term": "2013-2014",
   "committee_id": "NCC000009",
   "chamber": "upper",
```

```
    "state": "nc",
    "subcommittee": null,
    "committee": "Commerce",
    "position": "member",
    "type": "committee member"
   },
   {
    "term": "2013-2014",
    "committee_id": "NCC000100",
    "chamber": "upper",
    "state": "nc",
    "subcommittee": null,
    "committee": "Education / Higher Education",
    "position": "member",
    "type": "committee member"
   },
   {
    "term": "2013-2014",
    "committee_id": "NCC000073",
    "chamber": "upper",
    "state": "nc",
    "subcommittee": null,
    "committee": "Finance",
    "position": "member",
    "type": "committee member"
   },
   {
    "term": "2013-2014",
    "committee_id": "NCC000012",
    "chamber": "upper",
    "state": "nc",
    "subcommittee": null,
    "committee": "Health Care",
    "position": "member",
    "type": "committee member"
   },
   {
    "term": "2013-2014",
    "committee_id": "NCC000074",
    "chamber": "upper",
    "state": "nc",
    "subcommittee": null,
    "committee": "Judiciary I",
    "position": "member",
    "type": "committee member"
   },
   {
    "term": "2013-2014",
    "committee_id": "NCC000018",
    "chamber": "upper",
    "state": "nc",
    "subcommittee": null,
    "committee": "Rules and Operations of the Senate",
    "position": "member",
    "type": "committee member"
   }
  ],
  "level": "state",
```

```
  "url": "http://www.ncga.state.nc.us/gascripts/members/viewMember.pl?sChamber=Senate&
↪nUserID=267",
  "created_at": "2010-08-03 17:14:46",
  "nimsp_id": "9383",
  "chamber": "upper",
  "offices": [
   {
    "fax": null,
    "name": "Capitol Office",
    "phone": "(919) 715-6400",
    "address": "NC Senate\n16 W. Jones Street, Room 1113\n\nRaleigh, NC 27601-2808",
    "type": "capitol",
    "email": null
   }
  ],
  "suffixes": ""
 },
 {
  "last_name": "Hall",
  "updated_at": "2013-03-27 02:35:42",
  "sources": [
   {
    "url": "http://www.ncga.state.nc.us/gascripts/members/viewMember.pl?
↪sChamber=House&nUserID=679"
   }
  ],
  "full_name": "Duane Hall",
  "id": "NCL000282",
  "first_name": "Duane",
  "middle_name": "",
  "district": "11",
  "state": "nc",
  "party": "Democratic",
  "email": "Duane.Hall@ncleg.net",
  "leg_id": "NCL000282",
  "boundary_id": "sldl/nc-11",
  "+notice": null,
  "transparencydata_id": "07eff70ee51441d093b33667a2a6f877",
  "active": true,
  "photo_url": "http://www.ncga.state.nc.us/House/pictures/hiRes/679.jpg",
  "roles": [
   {
    "term": "2013-2014",
    "end_date": null,
    "district": "11",
    "chamber": "lower",
    "state": "nc",
    "party": "Democratic",
    "type": "member",
    "start_date": null
   },
   {
    "term": "2013-2014",
    "committee_id": "NCC000028",
    "chamber": "lower",
    "state": "nc",
    "subcommittee": null,
    "committee": "Appropriations",
```

```
    "position": "member",
    "type": "committee member"
   },
   {
    "term": "2013-2014",
    "committee_id": "NCC000035",
    "chamber": "lower",
    "state": "nc",
    "subcommittee": null,
    "committee": "Appropriations Subcommittee on Transportation",
    "position": "member",
    "type": "committee member"
   },
   {
    "term": "2013-2014",
    "committee_id": "NCC000082",
    "chamber": "lower",
    "state": "nc",
    "subcommittee": null,
    "committee": "Commerce and Job Development",
    "position": "member",
    "type": "committee member"
   },
   {
    "term": "2013-2014",
    "committee_id": "NCC000178",
    "chamber": "lower",
    "state": "nc",
    "subcommittee": null,
    "committee": "Commerce and Job Development Subcommittee on Alcoholic Beverage␣
↪Control",
    "position": "member",
    "type": "committee member"
   },
   {
    "term": "2013-2014",
    "committee_id": "NCC000168",
    "chamber": "lower",
    "state": "nc",
    "subcommittee": null,
    "committee": "Elections",
    "position": "member",
    "type": "committee member"
   },
   {
    "term": "2013-2014",
    "committee_id": "NCC000088",
    "chamber": "lower",
    "state": "nc",
    "subcommittee": null,
    "committee": "Government",
    "position": "member",
    "type": "committee member"
   },
   {
    "term": "2013-2014",
    "committee_id": "NCC000107",
    "chamber": "lower",
```

```
    "state": "nc",
    "subcommittee": null,
    "committee": "Homeland Security, Military, and Veterans Affairs",
    "position": "member",
    "type": "committee member"
   },
   {
    "term": "2013-2014",
    "committee_id": "NCC000172",
    "chamber": "lower",
    "state": "nc",
    "subcommittee": null,
    "committee": "Public Utilities and Energy",
    "position": "member",
    "type": "committee member"
   }
  ],
  "url": "http://www.ncga.state.nc.us/gascripts/members/viewMember.pl?sChamber=House&
↪nUserID=679",
  "created_at": "2013-01-03 19:15:14",
  "chamber": "lower",
  "offices": [
   {
    "fax": null,
    "name": "Capitol Office",
    "phone": "919-733-5755",
    "address": "NC House of Representatives\n16 W. Jones Street, Room 1019\n\nRaleigh,
↪ NC 27601-1096",
    "type": "capitol",
    "email": null
   }
  ],
  "suffixes": ""
 }
]
```


Committees

Committee Search Search committees by any of their attributes.

Committee Detail Get full detail for committee, including all members.

Committee Fields

The following fields are available on committee objects:

- `id` Open States assigned committee ID.
- `state` State abbreviation.
- `chamber` Chamber committee belongs to: 'upper', 'lower', 'joint'.
- `committee` Name of committee.
- `subcommittee` Name of subcommittee. (if null, object describes the `committee`)
- `parent_id` Committee id pointing to the parent committee if this is a subcommittee.

- `sources` List of URLs used in gathering information for this legislator.
- `created_at` The date that this object first appeared in our system.
- `updated_at` The date that this object was last updated in our system.
- `members` List of member objects, each has the following keys:
  - `name` Name of legislator as provided by state source.
  - `leg_id` Open States-assigned legislator id. (null if no match found).
  - `role` Member's role on the committee (e.g. 'chair', 'vice-chair', default role is 'member')

Methods

Committee Search

This method allows searching by a number of fields:

- `committee`
- `subcommittee`
- `chamber`
- `state`

Committee objects returned by this method do not include the list of members by default.

Example: *openstates.org/api/v1/committees/?state=dc*

Committee Detail

This method returns the full committee object given a committee id.

Example: *openstates.org/api/v1/committees/DCC000029/*

Examples

Committee Search

`openstates.org/api/v1/committees/?state=dc`


```
[
 { "level": "state",
  "created_at": "2011-11-09 02:43:35",
  "updated_at": "2013-03-27 03:23:42",
  "parent_id": null,
  "state": "dc",
  "subcommittee": null,
  "committee": "Finance and Revenue",
  "chamber": "upper",
  "id": "DCC000017" },
 { "level": "state",
  "created_at": "2011-11-09 02:43:35",
  "updated_at": "2013-03-06 02:18:33",
  "parent_id": null,
  "state": "dc",
```

```
  "subcommittee": null,
  "committee": "Subcommittee on Redistricting 2011",
  "chamber": "upper",
  "id": "DCC000025" },
 { "chamber": "upper",
  "created_at": "2013-01-07 21:05:11",
  "updated_at": "2013-03-27 03:23:42",
  "parent_id": null,
  "state": "dc",
  "subcommittee": null,
  "committee": "Business, Consumer and Regulatory Affairs",
  "id": "DCC000029" },
 { "level": "state",
  "created_at": "2011-11-09 02:43:35",
  "updated_at": "2013-03-27 03:23:41",
  "parent_id": null,
  "state": "dc",
  "subcommittee": null,
  "committee": "Human Services",
  "chamber": "upper",
  "id": "DCC000014" },
  ...truncated...
]
```


Committee Detail

`openstates.org/api/v1/committees/DCC000029/`


```
{
 "chamber": "upper",
 "committee": "Business, Consumer and Regulatory Affairs",
 "created_at": "2013-01-07 21:05:11",
 "id": "DCC000029",
 "members": [
  {
   "leg_id": "DCL000014",
   "role": "chairperson",
   "name": "Vincent Orange"
  },
  {
   "leg_id": "DCL000020",
   "role": "member",
   "name": "David Grosso"
  },
  {
   "leg_id": "DCL000007",
   "role": "member",
   "name": "Jim Graham"
  },
  {
   "leg_id": "DCL000002",
   "role": "member",
   "name": "Mary M. Cheh"
  },
  {
   "leg_id": "DCL000010",
```

```
    "role": "member",
    "name": "Yvette Alexander"
   }
  ],
  "parent_id": null,
  "sources": [ { "url": "http://dccouncil.us/committees/committee-on-business-consumer-
↪and-regulatory-affairs" } ],
  "state": "dc",
  "subcommittee": null,
  "updated_at": "2013-03-27 03:23:42"
}
```


Events

Events are not available in all states, to ensure that events are available check the `feature_flags` list in a states' *State Metadata*.

Event Search Search events by state and type.

Event Detail Get full detail for event.

Event Fields

The following fields are available on event objects:

- `id` Open States assigned event ID.
- `state` State abbreviation.
- `type` Categorized event type. ('committee:meeting' for now)
- `description` Description of event from state source.
- `documents` List of related documents.
- `location` Location if known, as given by state (is often just a room number).
- `when` Time event begins.
- `end` End time (null if unknown).
- `timezone` Timezone event occurs in (e.g. 'America/Chicago').
- `participants` List of participant objects, consisting of the following fields:
  - `chamber` Chamber of participant.
  - `type` Type of participants ('legislator', 'committee')
  - `participant` String representation of participant (e.g. 'Housing Committee', 'Jill Smith')
  - `id` Open States id for participant if a match was found (e.g. 'TXC000150', 'MDL000101')
  - `type` What role this participant played (will be 'host', 'chair', 'participant').
- `related_bills` List of related bills for this event. Comprised of the following fields:
  - `type` Type of relationship (e.g. 'consideration')
  - `description` Description of how the bill is related given by the state.
  - `bill_id` State's bill id (e.g. 'HB 273')

  - `id` Open States assigned bill id (e.g. 'TXB00001234')
- `sources` List of URLs used in gathering information for this legislator.
- `created_at` The date that this object first appeared in our system.
- `updated_at` The date that this object was last updated in our system.

Methods

Event Search

This method allows searching by a number of fields:

- `state`
- `type`

This method also allows specifying an alternate output format, by specifying `format=rss` or `format=ics`.

Example: *openstates.org/api/v1/events/?state=ca*

Event Detail

This method returns an event object given an event id.

Example: *openstates.org/api/v1/events/TXE00026474/*

Examples

Event Search

`openstates.org/api/v1/events/?state=tx`


```
[
 {
  "documents": [],
  "end": null,
  "description": "Special Purpose Districts",
  "state": "tx",
  "+agenda": "HOUSE OF REPRESENTATIVES NOTICE OF FORMAL MEETING \u00a0␣
↪COMMITTEE:\u00a0\u00a0 Special Purpose Districts\u00a0 TIME & DATE: During reading␣
↪and referral of bills Thursday, March 21, 2013\u00a0␣
↪PLACE:\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0 3W.9\u00a0␣
↪CHAIR:\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0 Rep. Dennis Bonnen\u00a0 \u00a0 \u00a0␣
↪Notice of this meeting was announced from the house floor.",
  "created_at": "2013-03-24 08:38:18",
  "when": "2013-03-21 05:00:00",
  "updated_at": "2013-03-24 08:38:18",
  "sources": [
   {
    "url": "http://www.capitol.state.tx.us/tlodocs/83R/schedules/html/
↪C4482013032100001.HTM"
   }
  ],
  "participants": [
   {
```

```
    "chamber": "lower",
    "participant_type": "committee",
    "participant": "Special Purpose Districts",
    "id": "TXC000150",
    "type": "host"
   },
   {
    "chamber": "lower",
    "participant_type": "legislator",
    "participant": "Rep. Dennis Bonnen",
    "id": "TXL000223",
    "type": "chair"
   }
  ],
  "session": "83",
  "location": "3W.9\u00a0 ",
  "related_bills": [],
  "timezone": "America/Chicago",
  "type": "committee:meeting",
  "id": "TXE00026474",
  "+chamber": "lower"
 },
 {
  "documents": [],
  "end": null,
  "description": "State Affairs",
  "state": "tx",
  "+agenda": "HOUSE OF REPRESENTATIVES NOTICE OF FORMAL MEETING \u00a0␣
↪COMMITTEE:\u00a0\u00a0 State Affairs\u00a0 TIME & DATE: During reading and referral␣
↪of bills Thursday, March 21, 2013\u00a0 PLACE:\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0␣
↪Agricultural Museum, 1W.14\u00a0 CHAIR:\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0 Rep.␣
↪Byron Cook\u00a0 \u00a0 Notice of this meeting was announced from the House floor.",
  "created_at": "2013-03-24 08:38:18",
  "when": "2013-03-21 05:00:00",
  "updated_at": "2013-03-24 08:38:18",
  "sources": [
   {
    "url": "http://www.capitol.state.tx.us/tlodocs/83R/schedules/html/
↪C4502013032100001.HTM"
   }
  ],
  "participants": [
   {
    "chamber": "lower",
    "participant_type": "committee",
    "participant": "State Affairs",
    "id": "TXC000022",
    "type": "host"
   },
   {
    "chamber": "lower",
    "participant_type": "legislator",
    "participant": "Rep. Byron Cook",
    "id": "TXL000236",
    "type": "chair"
   }
  ],
  "session": "83",
```

```
  "location": "Agricultural Museum, 1W.14\u00a0 ",
  "related_bills": [],
  "timezone": "America/Chicago",
  "type": "committee:meeting",
  "id": "TXE00026476",
  "+chamber": "lower"
 },
 {
  "documents": [],
  "end": null,
  "description": "Defense & Veterans' Affairs",
  "type": "committee:meeting",
  "created_at": "2013-03-15 07:37:08",
  "related_bills": [
   {
    "type": "consideration",
    "description": "Bill up for discussion",
    "bill_id": "HB 846",
    "id": "TXB00024869"
   },
   {
    "type": "consideration",
    "description": "Bill up for discussion",
    "bill_id": "HB 1348",
    "id": "TXB00025984"
   },
   {
    "type": "consideration",
    "description": "Bill up for discussion",
    "bill_id": "HB 1832",
    "id": "TXB00026956"
   },
   {
    "type": "consideration",
    "description": "Bill up for discussion",
    "bill_id": "HB 1939",
    "id": "TXB00027260"
   },
   {
    "type": "consideration",
    "description": "Bill up for discussion",
    "bill_id": "HB 2387",
    "id": "TXB00028147"
   },
   {
    "type": "consideration",
    "description": "Bill up for discussion",
    "bill_id": "HB 2392",
    "id": "TXB00028152"
   },
   {
    "type": "consideration",
    "description": "Bill up for discussion",
    "bill_id": "HB 2071",
    "id": "TXB00027470"
   }
  ],
  "when": "2013-03-21 13:00:00",
```

```
  "updated_at": "2013-03-21 08:03:49",
  "sources": [
   {
    "url": "http://www.capitol.state.tx.us/tlodocs/83R/schedules/html/
↪C3052013032108001.HTM"
   }
  ],
  "state": "tx",
  "session": "83",
  "location": "E2.012\u00a0 ",
  "participants": [
   {
    "chamber": "lower",
    "participant_type": "committee",
    "participant": "Defense & Veterans' Affairs",
    "id": "TXC000058",
    "type": "host"
   },
   {
    "chamber": "lower",
    "participant_type": "legislator",
    "participant": "Rep. Jos\u00e9 Men\u00e9ndez",
    "id": "TXL000312",
    "type": "chair"
   }
  ],
  "timezone": "America/Chicago",
  "+agenda": "** REVISION **HOUSE OF REPRESENTATIVES NOTICE OF PUBLIC HEARING \u00a0␣
↪COMMITTEE:\u00a0\u00a0 Defense & Veterans' Affairs\u00a0 TIME & DATE: 8:00 AM,␣
↪Thursday, March 21, 2013\u00a0 PLACE:\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0 E2.
↪012\u00a0 CHAIR:\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0 Rep. Jos\u00e9␣
↪Men\u00e9ndez\u00a0 \u00a0 HB 846\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0 Lucio␣
↪III Relating to additional periods of possession of or access to a child after␣
↪conclusion of a parent's military deployment. HB␣
↪1348\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0␣
↪Men\u00e9ndez\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u
↪Relating to the taxation of certain tangible personal property located inside a␣
↪defense base development authority. HB 1832\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0␣
↪Miller, Rick\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0␣
↪Relating to granting certain local governments general zoning authority around␣
↪certain military facilities; providing a penalty. HB␣
↪1939\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0␣
↪Orr\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a
↪Relating to a veteran's employment preference for employment with a public entity␣
↪or public work of this state. HB 2387\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0␣
↪Men\u00e9ndez\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u
↪Relating to the taxation of certain tangible personal property located inside a␣
↪defense base development authority. HB 2392\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0␣
↪Men\u00e9ndez\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u
↪Relating to the mental health program for veterans. \u00a0 \u00a0 Bills deleted␣
↪after last posting: HB 2071 HCR 69 \u00a0␣
↪**\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0 See Committee Coordinator for␣
↪previous versions\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0 ** of the␣
↪schedule, if applicable. NOTICE OF ASSISTANCE AT PUBLIC MEETINGS Persons with␣
↪disabilities who plan to attend this meeting and who may need assistance, such as a␣
↪sign language interpreter, are requested to contact Stacey Nicchio at (512) 463-
↪0850, 72 hours prior to the meeting so that appropriate arrangements can be made.␣
↪\u00a0 To find information about electronic witness registration for a public␣
↪hearing and to create a profile to be used when registering as a witness, please␣
↪visit www.house.state.tx.us/resources/. Registration must be performed the day of␣
↪the meeting and within the Capitol Complex.",
```

```
  "id": "TXE00026387",
  "+chamber": "lower"
 },
 ...truncated...
]
```


Event Detail

openstates.org/api/v1/event/TXE00026474/


```
{
 "+agenda": "HOUSE OF REPRESENTATIVES NOTICE OF FORMAL MEETING \u00a0␣
↪COMMITTEE:\u00a0\u00a0 Special Purpose Districts\u00a0 TIME & DATE: During reading␣
↪and referral of bills Thursday, March 21, 2013\u00a0␣
↪PLACE:\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0 3W.9\u00a0␣
↪CHAIR:\u00a0\u00a0\u00a0\u00a0\u00a0\u00a0 Rep. Dennis Bonnen\u00a0 \u00a0 \u00a0␣
↪Notice of this meeting was announced from the house floor.",
 "+chamber": "lower",
 "created_at": "2013-03-24 08:38:18",
 "description": "Special Purpose Districts",
 "documents": [],
 "end": null,
 "id": "TXE00026474",
 "location": "3W.9\u00a0 ",
 "participants": [
  {
   "chamber": "lower",
   "participant_type": "committee",
   "participant": "Special Purpose Districts",
   "id": "TXC000150",
   "type": "host"
  },
  {
   "chamber": "lower",
   "participant_type": "legislator",
   "participant": "Rep. Dennis Bonnen",
   "id": "TXL000223",
   "type": "chair"
  }
 ],
 "related_bills": [],
 "session": "83",
 "sources": [
  {
   "url": "http://www.capitol.state.tx.us/tlodocs/83R/schedules/html/
↪C4482013032100001.HTM"
  }
 ],
 "state": "tx",
 "timezone": "America/Chicago",
 "type": "committee:meeting",
 "updated_at": "2013-03-24 08:38:18",
 "when": "2013-03-21 05:00:00"
}
```

Districts

District Search List districts for state (and optionally filtered by chamber).

District Boundary Lookup Get geographic boundary for a district.

Methods

District Search

The district search method requires a `state` and can optionally also take a `chamber` as part of the URL.

The method returns a list of district objects with the following fields:

- `abbr` State abbreviation.
- `boundary_id` boundary_id used in *District Boundary Lookup*
- `chamber` Whether this district belongs to the upper or lower chamber.
- `id` A unique ID for this district (separate from boundary_id).
- `legislators` List of legislators that serve in this district. (may be more than one if `num_seats` > 1)
- `name` Name of the district (e.g. '14', '33A', 'Fifth Suffolk')
- `num_seats` Number of legislators that are elected to this seat. Generally one, but will be 2 or more if the seat is a multi-member district.

Example: *openstates.org/api/v1/districts/nc/lower/*

District Boundary Lookup

This method returns an full district object, including the boundary given a `boundary id`.

The returned object has the following fields:

- `abbr` State abbreviation.
- `bbox` A bounding box composed of a list of two (long, lat) points. The first point is the upper left corner, and the second point is the lower right.
- `boundary_id` boundary_id for this boundary.
- `chamber` Whether this district belongs to the upper or lower chamber.
- `id` A unique ID for this district (separate from boundary_id).
- `name` Name of the district (e.g. '14', '33A', 'Fifth Suffolk')
- `num_seats` Number of legislators that are elected to this seat. Generally one, but will be 2 or more if the seat is a multi-member district.
- `region` A dictionary of the following values:
  - `center_lat` Center latitude of the bounding box.
  - `center_lon` Center longitude of the bounding box.
  - `lat_delta` Equivalent to max(`latitude`)-min(`latitude`)
  - `lon_delta` Equivalent to max(`longitude`)-min(`longitude`)
- `shape` List of polygons, each of which is a GeoJSON-like list of coordinates describing a single polygon.

Example: *openstates.org/api/v1/districts/boundary/sldl/nc-120/*

Examples

District Search

`openstates.org/api/v1/districts/nc/lower/`


```
[
 { "abbr": "nc",
  "boundary_id": "sldl/nc-1",
  "chamber": "lower",
  "id": "nc-lower-1",
  "legislators": [
   { "full_name": "Bob Steinburg", "leg_id": "NCL000302" }
  ],
  "name": "1",
  "num_seats": 1 },
 { "abbr": "nc",
  "boundary_id": "sldl/nc-12",
  "chamber": "lower",
  "id": "nc-lower-12",
  "legislators": [
   { "full_name": "George Graham", "leg_id": "NCL000281" }
  ],
  "name": "12",
  "num_seats": 1 },
 { "abbr": "nc",
  "boundary_id": "sldl/nc-13",
  "chamber": "lower",
  "id": "nc-lower-13",
  "legislators": [
   { "full_name": "Pat McElraft", "leg_id": "NCL000137" }
  ],
  "name": "13",
  "num_seats": 1 },
 { "abbr": "nc",
  "boundary_id": "sldl/nc-14",
  "chamber": "lower",
  "id": "nc-lower-14",
  "legislators": [
   { "full_name": "George G Cleveland", "leg_id": "NCL000076" }
  ],
  "name": "14",
  "num_seats": 1 },
 { "abbr": "nc",
  "boundary_id": "sldl/nc-15",
  "chamber": "lower",
  "id": "nc-lower-15",
  "legislators": [
   { "full_name": "Phil R Shepard", "leg_id": "NCL000221" }
  ],
  "name": "15",
  "num_seats": 1 },
 ... truncated ...
]
```

District Boundary Lookup

`openstates.org/api/v1/districts/boundary/sldl/nc-120/`


```
{
 "abbr": "nc",
 "bbox": [
  [ 34.986592, -84.321869 ],
  [ 35.466558, -83.108571 ]
 ],
 "boundary_id": "sldl/nc-120",
 "chamber": "lower",
 "id": "nc-lower-120",
 "name": "120",
 "num_seats": 1,
 "region": {
  "center_lat": 35.226575,
  "center_lon": -83.71522,
  "lat_delta": 0.47996599999999745,
  "lon_delta": 1.2132980000000089
 },
 "shape": [
  [
   [
    [ -84.321797, 34.988965 ],
    [ -84.308201, 35.092843 ],
    [ -84.30696, 35.106162 ],
    [ -84.297721, 35.169478 ],
    [ -84.294723, 35.185594 ],
    [ -84.29024, 35.225572 ],
    [ -84.289921, 35.225585 ],
    [ -84.290061, 35.225257 ],
    [ -84.289621, 35.224677 ],
    [ -84.288516, 35.224391 ],
    [ -84.28712, 35.224877 ],
    [ -84.28512, 35.226577 ],
    [ -84.28322, 35.226577 ],
    [ -84.28152, 35.229277 ],
    [ -84.27792, 35.231477 ],
    [ -84.27702, 35.233177 ],
    [ -84.27662, 35.233277 ],
    ... truncated ..
   ],
   ... truncated ...
  ]
 ]
}
```

Methods

Method	URL pattern	Description
Metadata Overview	/metadata/	Get list of all states with data available and basic metadata about their status.
State Metadata	/metadata/*state*/	Get detailed metadata for a particular state.
Bill Search	/bills/	Search bills by (almost) any of their attributes, or full text.
Bill Detail	/bills/*state*/*session*/*bill_id*/	Get full detail for bill, including any actions, votes, etc.
Legislator Search	/legislators/	Search legislators by their attributes.
Legislator Detail	/legislators/*leg_id*/	Get full detail for a legislator, including all roles.
Geo Lookup	/legislators/geo/?lat=latitude&long=longitude	Lookup all legislators that serve districts containing a given point.
Committee Search	/committees/	Search committees by any of their attributes.
Committee Detail	/committees/*committee_id*/	Get full detail for committee, including all members.
Event Search	/events/	Search events by state and type.
Event Detail	/event/*event_id*/	Get full detail for event.
District Search	/districts/*state*/[*chamber*/]	List districts for state (and optionally filtered by chamber).
District Boundary Lookup	/districts/boundary/*boundary_id*/	Get geographic boundary for a district.

Requesting A Custom Fieldset

On essentially every method in the API it is possible to specify a custom subset of fields on an object by specifying a `fields` parameter.

There are two use cases that this functionality aims to serve:

First, if you are writing an application that loads a lot of data but only uses some of it, specifying a limited subset of fields can reduce response time and bandwidth. We've seen this approach be particuarly useful for mobile applications where bandwidth is at a premium.

An example would be a legislator search with `fields=first_name,last_name,leg_id` specified. All legislator objects returned will only have the three fields that you requested.

Second, you can actually specify a set of fields that includes fields excluded in the default response.

For instance, if you are conducting a bill search, it typically does not include sponsors, though many sites may wish to use sponsor information without making a request for the full bill (which is typically much larger as it includes versions, votes, actions, etc.).

A bill search that specifies `fields=bill_id,sponsors,title,chamber` will include the full sponsor listing in addition to the standard bill_id, title and chamber fields.

Extra Fields

You may notice that the fields documented are sometimes a subset of the fields actually included in a response.

Many times as part of our scraping process we take in data that is available for a given state and is either not available or does not have an analog in other states. Instead of artificially limiting the data we provide to the smallest common subset we make this extra data available.

To make it clear which fields can be relied upon and which are perhaps specific to a state or subset of states we prefix non-standard fields with a +.

If you are using the API to get data for multiple states, it is best to restrict your usage to the fields documented here. If you are only interested in data for a small subset of our available states it might make sense to take a more in depth look at the API responses for the state in question to see what extra data we are able to provide.

CHAPTER 2

Contributing to Open States

Open States' origin is as a community-driven project, and is only possible because of the sustained effort of dozens of volunteers.

The guides below will walk you through how we work, and various ways you can contribute:

Code of Conduct

Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks

- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at contact@openstates.org. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at http://contributor-covenant.org/version/1/4.

Communication

When joining a new community, it can be tough to figure out *where* to ask questions, provide feedback, or help out. Don't worry! As long as you're respectful and follow our *Code of Conduct*, we're happy to have you!

Here are some guidelines regarding the best way to get in touch or contribute. Do note that Open States is a volunteer-powered project, and all of the core developers have day jobs; we're excited to talk to you, but will be most communicative outside of East Coast business hours.

Recommendations

Want to ask a general question, have a conversation, or listen in to the community?

The best place is our Slack Channel. This is the main place for Open States discussion. The core team and many other contributors are present there, and we're usually able to answer questions in a timely fashion.

Have a private or financial question, or a security concern?

Email contact@openstates.org; only the administrative team can see these.

Have you found an error or issue in the Open States data?

File an issue on our bug tracker. And before you do, quickly check whether anyone else there has already reported the same bug.

Have a technical issue not related to the data itself?

Try to find the appropriate repository in our GitHub organization, and file an issue there. For example:

- openstates.org: https://github.com/openstates/openstates.org/issues/
- our documentation: https://github.com/openstates/documentation/issues/

Have a feature request, or something else that doesn't fit here?

We use the openstates/meta repository to track feature requests and big-picture questions. If in doubt, it is OK to post to this repo. We may move your ticket to the appropriate place.

Discouraged Methods of Communication

Avoid using these channels to get in touch with us:

Personal email addresses of Open States developers

Please refrain from contacting any of the developers directly outside of Slack, unless we ask you to do so.

Twitter (or any other social media)

We mainly use the @openstates twitter account to make announcements, and don't have the resources to provide technical support or other feedback on Twitter.

The old Google Group

Because of spam, every post to the group has to be moderated, which makes it a much slower and less effective way to get in touch.

We're considering deprecation of the group sometime in 2017.

Start Contributing to Open States

Note: This document is very much a work-in-progress, feel free to suggest contributions.

Scrapers are at the core of what Open States does, each state requires several custom scrapers designed to extract bills, legislators, committees, votes, and events from the state website. All together there are around 200 scrapers, each one essentially independent, which means that there is always more work to do, but plenty of prior work to learn from.

Code of Conduct

Open States is a project that can only exist because of the fantastic community that has worked on it over the years. In the interest of keeping this community a healthy and welcoming place we have a *Code of Conduct* and encourage you to familiarize yourself with it.

Prerequisites

This guide assumes a basic familiarity with:

- using the command line
- git
- Python

No worries if you aren't an expert though, we'll walk you through the steps. And as for Python, if you've written other languages like Javascript or Ruby you'll probably be just fine. Don't be afraid to *ask for help* either!

Getting Started

First thing you will need to do is get a working development environment on your local machine. We'll do this using Docker. No worries if you aren't familiar with Docker, you'll barely have to touch it.

Step 1) Install Docker (and docker-compose) if not already installed on your local system.

- On OSX: Docker for Mac is perhaps the easiest way.
- On Windows: Docker for Windows
- On Linux: Use your package manager of choice or follow Docker's instructions.
- Generic instructions from Docker.

Step 2) Ensure that Docker (and docker-compose) are installed locally and check their versions:

```
$ docker --version
Docker version 17.03.0-ce, build 60ccb22
$ docker-compose --version
docker-compose version 1.11.2, build dfed245
```

Of course, your versions may be newer. The minimum required versions for Open States are:

- 1.9.0 of Docker
- 1.10.0 of Docker Compose

Step 3) We'll fork and clone the main Open States scraper repository.

- Visit https://github.com/openstates/openstates and click the 'Fork' button.
- Clone your fork using your tool of choice or the command line:

```
$ git clone git@github.com:yourname/openstates.git
Cloning into 'openstates'...
```

At this point you'll have a local `openstates` directory. Let's go ahead and look at it:

```
$ cd openstates
$ ls
AUTHORS            README.rst         experimental       requirements.txt
Dockerfile         billy_settings.py  manual_data        scripts
LICENSE            docker-compose.yml openstates         setup.py
```

There are a few top level text files, some docker files, which we'll come back to shortly, and some directories. The directory we care about is the one called `openstates`.:

```
$ ls openstates
__init__.py dc          in          mn          nj          pr          va
ak          de          ks          mo          nm          ri          vi
al          fl          ky          ms          nv          sc          vt
ar          ga          la          mt          ny          sd          wa
az          hi          ma          nc          oh          tn          wi
ca          ia          md          nd          ok          tx          wv
co          id          me          ne          or          ut          wy
ct          il          mi          nh          pa          utils
```

This directory is a python module with 50+ subpackages, one for each state.

Let's look inside one:

```
$ ls openstates/nc
__init__.py    bills.py       committees.py  people.py      votes.py
```

Some will differ a bit, but all will have `__init__.py`, `bills.py`, and either `legislators.py` or `people.py`. These are the NC scrapers that collect these objects.

Step 4) Let's finish setting up our environment by creating the database:

```
$ docker-compose up database
```

The output of the '... up database...' command should end with:: NETWORK [thread1] waiting for connections on port 27017

If, instead, these commands fail, check that your copy of docker-compose is a recent vintage.

Note: If you encounter an error like:

```
database_1        | chown: changing ownership of '/dev/stdout': Permission denied
database_1        | chown: changing ownership of '/dev/stderr': Permission denied
```

It is likely related to Docker Issue #31243 which you can work around by adding `tty: true` in your docker-compose's `database` entry.

At this point we have two docker images:

database A MongoDB database that we're going to use to store our scraped data.

openstates The image that will run our scrapers.

And we're ready to go!

Running Our First Scraper

Step 5) Open a new terminal tab in preparation for the '... run openstates ...' command.

Step 6) Choose a state.

Step 7) Let's run <your state's> legislator scraper (substitute your state for 'nc' below)

```
$ docker-compose run --rm openstates nc --legislators --fast
```

The parameters you pass after `docker-compose run --rm openstates` are passed to `billy-update`. Here we're saying that we're running NC's scrapers, just want to run the legislators scraper, and that we want to do it in "fast mode." A full description of `billy-update` is available in the billy docs.

So, `billy-update` kicks off a full scrape of NC's current legislators. You'll start seeing things like:

```
18:15:16 INFO billy: billy-update abbr=nc
    actions=scrape,import,report
    types=legislators
    sessions=2017
    terms=2017-2018
18:15:18 INFO scrapelib: GET - http://www.ncga.state.nc.us/gascripts/members/
↪memberListNoPic.pl?sChamber=Senate
18:15:19 INFO scrapelib: GET - http://www.ncga.state.nc.us/gascripts/members/
↪viewMember.pl?sChamber=Senate&nUserID=392
18:15:20 INFO billy: Save person John M. Alexander, Jr.
18:15:21 INFO scrapelib: GET - http://www.ncga.state.nc.us/gascripts/members/
↪viewMember.pl?sChamber=Senate&nUserID=396
18:15:22 INFO billy: Save person Deanna Ballard
18:15:22 INFO scrapelib: GET - http://www.ncga.state.nc.us/gascripts/members/
↪viewMember.pl?sChamber=Senate&nUserID=369
18:15:23 INFO billy: Save person Chad Barefoot
```

The first thing is billy's *run plan*, what it is going to try to scrape. This is presented as a sanity check, and each of these values can be controlled by different command line parameters. In this case we see we're running the *scrape*, *import*, and *report* actions for nc legislators for 2017-2018. The scraper chose the most recent available session/term for us.

Depending on the scraper you run, this part takes a while. Some bill scrapers can take hours to run, but most legislator scrapers are a few minutes.

At the end of the scrape you should see a message like:

```
18:19:18 INFO billy: Finished importing 169 legislator files.
```

This means that the data is now in the database. Congratulations, you just ran your first state scrape!

Step 8) To access the data you just fetched, you can connect to the database as follows:

```
$ docker-compose run --entrypoint mongo database mongodb://database
```

This loads the mongodb shell. You may close the mongo connection with:: > quit()

You can also view the data in the `data` directory of the project root.

Note: It is of course possible that the scrape fails. If so, there's a good chance that isn't your fault, especially if it starts to run and then errors out. Scrapers do break, and there's no guarantee North Carolina didn't change their legislator page yesterday, breaking our tutorial here.

If that's the case and you think the issue is with the scraper, feel free to get in touch with us or file an issue.

Next Steps

At this point you're ready to run scrapers and contribute fixes.

Right now the most important task in front of us is converting scrapers to pupa, see *Converting Scrapers to pupa* and consider helping us out today!

Getting Help

Right now the best way to get help is to join our Slack, plenty of the core team and other contributors are around to answer any questions you may have.

Converting Scrapers to pupa

Note: This document is a work-in-progress; if any part of it is unclear, suggest changes or improvements.

As of early 2017, we've embarked on a process to switch away from our legacy backend (billy) that has been in use since 2009, to a more modern backend (pupa) based on the Open Civic Data specification and tools. We've written in a few places why this change is important and worth our time:

- https://github.com/openstates/meta/wiki/2017-Roadmap#pupa-ization
- https://blog.openstates.org/post/whats-next-2017/

This task will require updates to every single one of our scrapers. Given that this is such a big task, and will enable many more converting scrapers from billy to pupa is one of the best ways to help out on Open States right now. Follow this guide and start converting a state!

Before You Start

If you haven't already, you should read our *Start Contributing to Open States* guide. It's important to know how scrapers are run, before starting to convert them.

It's also important to make sure that the state you've chosen to convert from billy to pupa hasn't already been converted! This is tracked in ticket #1442. If your state of choice isn't available, consider picking another state, or fixing some Open States bugs.

Begin

Comment on the aforementioned tracking ticket so that no one else accidentally works on this state as well, duplicating effort. And once you have a Git branch with some conversion work in it, open a WIP PR (Work-in-Progress Pull Request) against the Open States repository, so that others can follow along.

Converting Metadata

For example purposes, let's look at what it takes to convert North Carolina. Going forward, you can replace `nc` with the abbreviation for your selected state.

Each state has static metadata found in `openstates/{{state}}/__init__.py`. The first step will be to convert this metadata from the billy format to the new pupa format.

1. Start by moving the existing metadata to our new `billy_metadata/` directory; until all states are in Pupa format, we're going to need to keep the billy-specific metadata around:

```
$ git mv openstates/nc/__init__.py billy_metadata/nc.py
```

2. Edit `billy_metadata/nc.py`:

Delete everything in the module besides the `metadata` dictionary and any imports it requires. You temporarily may want to leave `session_list` around as well, as we'll be using it in the next step.

Example diff: NC billy_metadata

3. Create a new `openstates/nc/__init__.py`:

   There's a script to help with this step. It isn't guaranteed to work perfectly, but should at least provide a good starting point:

   ```
   $ ./scripts/convert_metadata.py nc > openstates/nc/__init__.py
   ```

   Then, set the `url` property inside to a valid URL representing the state's government. You may need to modify the file further, such as indicating the number of seats in the upper and lower houses of your state.

   Delete the *session_list* function from *billy_metadata/nc.py* add it to the jurisdiction subclass in the newly created *openstates/nc/__init__.py* file, renamed to *get_session_list*.

   Example diff: updated NC metadata

At this point you should have a fairly complete OCD jurisdiction defined. Next, we'll move on to converting the legislator scraper.

Converting Legislators

We won't be able to test our pupa metadata file until we write a pupa scraper, so let's do that!

For a state in the billy framework, we have a `legislators.py` file that contains a scraper instantiated from `billy.scrape.legislators.Legislator`. This scraper captures and saves `Legislator` objects.

In pupa, this scraper is called `people` scraper instead. (This is because OCD can easily model individuals who aren't members of a legislature.)

As you'll see, pupa scrapers `yield` scraped objects, whereas a billy scraper would call `save_legislator`; `yield` and `yield from` expose Python 3's powerful generator and subroutine capabilities.

Before diving in, it's helpful to look over the docs for billy legislator scrapers and pupa person scrapers.

1. Rename the scraper file:

   ```
   $ git mv openstates/nc/legislators.py openstates/nc/people.py
   ```

2. Open `people.py` and update the `import` statement:

   At the top of the file, you'll see something like:

   ```
   from billy.scrape.legislators import LegislatorScraper, Legislator
   ```

   We instead want:

   ```
   from pupa.scrape import Person, Scraper
   ```

   (pupa doesn't have different `Scraper` subclasses.)

   Also, rename the file's instantiated scraper (so that it refers to `Person` rather than `Legislator`), and make it a subclass of `Scraper` rather than `LegislatorScraper`. Using the North Carolina example, you would convert this:

   ```
   class NCLegislatorScraper(LegislatorScraper):
   ```

to this:

```
class NCPersonScraper(Scraper):
```

Note that if your class also subclasses from something else like `LXMLMixin`, do not remove that. For example, if you had `class NCLegislatorScraper(LegislatorScraper, LXMLMixin):`, then you would change it to `class NCPersonScraper(Scraper, LXMLMixin):`.

3. Update the `scrape` method's signature:

   In billy scrapers, the `scrape` method signature is `scrape(term, chambers)`, and serves as an entrypoint for the scraper class.

   pupa scrapers also use a `scrape` method as an entrypoint, but the parameters are all optional.

   Because most legislator scrapers only scrape the current session, we'll drop the `term` argument, and the `chambers` argument can be made into an optional `chamber` argument.

   The NC scraper already had a `scrape_chamber` method that was invoked by the `scrape` method. So, we updated our `scrape` method to dispatch like this:

   ```
   def scrape(self, chamber=None):
       if chamber:
           yield from self.scrape_chamber(chamber)
       else:
           yield from self.scrape_chamber('upper')
           yield from self.scrape_chamber('lower')
   ```

   pupa `scrape` methods (which are generators) must `yield` objects. Since the NC scraper's `scrape_chamber` method (also a generator) will collect and `yield` the People objects initially, the `scrape` method must `yield from` that generator itself.

4. Update the portion of the code that creates and saves `Legislator` objects:

   The billy scrapers create `Legislator` objects, and then call `self.save_legislator`. We'll need to turn `self.save_legislator` into a `yield` of `Person` objects.

   This change is typically minimal; there's a lot of code in billy legislator scrapers, but very little of it should need to be edited for the purposes of pupa.

   Instead of instantiating `Legislator` objects, instantiate `Person` objects. You should also name the variable that will hold your `Person` as `person`, whereas your `Legislator` was probably assigned to `leg` or `legislator`. Then, in all remaining code, replace `leg`/`legislator` with `person`.

   All arguments passed to `Person` need to be named; some states' old scrapers do not assign names to all arguments, so you will need to add argument names in those cases. Also, some named arguments have changed names. Your `Person` should only take these five arguments:

   - `primary_org` (may have been `chamber`)
   - `district`
   - `name` (may have been `full_name`)
   - `party`
   - `image` (may have been `photo_url`)

   Using the `chamber` to `primary_org` change as an example, your instantiation of the `Legislator` will probably say either `chamber=chamber` or just `chamber`, but in either case

should be changed to `primary_org=chamber` when instantiating the `Person`. Note that there is no need to change the variable name earlier in the code.

Instead of passing `url` as an argument, add any such links with `Person.add_link`.

`term` should no longer be given as an argument. Any extra arguments that were given to `Legislator` (besides the five listed above) can be placed in `Person`'s `extras` dictionary. For example, if `Legislator` was given a `town_represented`, you would instead do something like this:

```
person.extras['town_represented'] = town_represented
```

For contact information, instead of using `add_office`, you'll use `Person`'s `add_contact_detail` method. For example, adding a district office's phone number might look something like this:

```
person.add_contact_detail(type='voice', value=contact_info['phone'], note=
↪'District Office')
```

Note that `contact_info['phone']` above should be replaced with wherever that phone number was stored earlier in the code. The `type` comes from the Popolo standard.

Instead of `self.save_legislator(Legislator)` from billy, end with `yield person`. (Make sure that any function that creates `Person` objects outside of `scrape` is invoked by `scrape` using `yield from`, as described above.)

Again, it might be a good idea to look over the docs for billy legislator scrapers and pupa person scrapers.

Since you're also switching from Python 2 (billy) to Python 3 (pupa), you may need to make syntax changes to the module. For instance, if `Dict.iteritems()` is used anywhere, it would have to be replaced by `Dict.items()`. Also, `xrange` will need to be replaced by `range`.

At this point, your person scraper should essentially be converted.

Example diff: converted legislator scraper (there may be significant differences between the North Carolina example and your state)

5. Revisiting the metadata:

   We now need to make one small change to the metadata (ie, the `__init__.py` file) to let pupa know about our person scraper. Import our new scraper at the top of `openstates/nc/__init__.py`:

   ```
   from .people import NCPersonScraper
   ```

   And within the Jurisdiction object, update the `scrapers` dictionary to look like:

   ```
   scrapers = {
       'people': NCPersonScraper,
   }
   ```

6. Running your first scraper:

   Now let's try giving it a run:

   ```
   $ docker-compose run scrape nc
   ```

   This runs pupa scrapers for the state. A second script is then executed, back-porting the scraped pupa data to billy format; since the API and website currently rely on the billy format, this is necessary during the transition off of billy.

You'll probably see output like:

```
no pupa_settings on path, using defaults
nc (scrape)
  people: {}
Not checking sessions...
15:35:05 INFO pupa: save jurisdiction North Carolina as jurisdiction_ocd-jurisdiction-
↪country:us-state:nc-government.json
15:35:05 INFO pupa: save organization North Carolina General Assembly as organization_
↪6ecadcc4-0122-11e7-91f7-0242ac130003.json
15:35:05 INFO pupa: save organization Senate as organization_6ecae228-0122-11e7-91f7-
↪0242ac130003.json
15:35:05 INFO pupa: save post 1 as post_6ecb36e2-0122-11e7-91f7-0242ac130003.json
15:35:05 INFO pupa: save post 2 as post_6ecb3840-0122-11e7-91f7-0242ac130003.json
15:35:05 INFO pupa: save post 3 as post_6ecb3976-0122-11e7-91f7-0242ac130003.json
15:35:05 INFO pupa: save post 4 as post_6ecb3ab6-0122-11e7-91f7-0242ac130003.json
```

The `people: {}` line describes what type of data pupa is trying to scrape, that it has found your Person scraper, and that it is running without any arguments.

Next, you see the line `Not checking sessions...`, which we'll revisit later.

If all goes well, the scraper will run for a while, writing JSON objects to the `_data` directory as it goes.

Finally, you'll see output like:

```
nc (scrape)
  people: {}
jurisdiction scrape:
  duration:  0:00:00.561228
  objects:
    jurisdiction: 1
    organization: 5
    post: 170
people scrape:
  duration:  0:00:03.910275
  objects:
    membership: 340
    person: 170
```

This is the result of the scrape, including the metadata and person objects that were successfully collected.

Once that is done you'll see the to-billy conversion begin, ultimately ending in some lines like:

```
15:43:34 INFO billy: billy-update abbr=nc
    actions=import,report
    types=bills,legislators,votes,committees,alldata
    sessions=2017
    terms=2017-2018
15:43:35 INFO billy: Finished importing 170 legislator files.
15:43:35 INFO billy: imported 0 vote files
15:43:35 INFO billy: imported 0 bill files
15:43:35 INFO billy: imported 0 committee files
```

The import part to check is the `{{n}} legislator files`, which ought to match the number of person objects reported by pupa.

Once you get to this point, you have successfully converted a scraper to pupa! Congratulations, and thank you! Let's make sure your hard work gets integrated.

Creating Your Pull Request

Once you have this work done, go ahead and let us know so that we can avoid duplicating effort.

The preferred way to do this is to open a work-in-progress PR, naming your PR something like `[WIP] Convert {{state}} to pupa`. A helpful guide to making PRs with GitHub is here: https://help.github.com/articles/creating-a-pull-request/

Someone from the team will review the PR and possibly request that you make some minor fixes, but no matter the status your work will be helpful. If you'd like to continue on, *Converting Scrapers to pupa (continued)* has information on converting the remaining types of scrapers.

Converting Scrapers to pupa (continued)

Picking up where we left off in *Converting Scrapers to pupa*, we'll now look at how to convert committee, bill, vote, and event scrapers from billy to pupa.

Converting Committees

There may not be a committee scraper for your selected state. But good news if there is: committee scrapers are straightforward to convert.

In pupa, committees are just a type of `Organization`. You've already seen `Organization` objects briefly, in the `__init__.py` pupa metadata file; the legislative branch itself is a pupa `Organization`, as are its individual chambers.

Here's how to convert the scraper:

1. Update the imports and class definition:

```
# old
from billy.scrape.committees import CommitteeScraper, Committee

# new
from pupa.scrape import Scraper, Organization
```

And:

```
# old
class NCCommitteeScraper(CommitteeScraper):
    jurisdiction = 'nc'

# new
class NCCommitteeScraper(Scraper):
```

2. Add the new class name to metadata file:

In `openstates/nc/__init__.py`, add:

```
from .committees import NCCommitteeScraper
```

And add the scraper instance to the `scrapers` in that file:

```
scrapers = {
    'people': NCPersonScraper,
```

```
    'committees': NCCommitteeScraper,
}
```

3. Update the `scrape` method:

   As we saw with the people scraper, the `scrape` method no longer has required parameters. It should no longer take a `term` parameter, and `chambers` should become optional.

   Example diff:

   Additionally, the `scrape` method should now `yield` objects instead of calling `save_committee`.

   Note: Recall that, as before, if the `scrape` method dispatches other methods, it should call them with `yield from`, and they should be converted to `yield` objects as needed.

   ---

4. Update `Committee` references to `Organization`:

   The old constructor was `Committee(chamber, committee, subcommittee)`. Under pupa, constructing a committee looks like:

```
committee = Organization(
    name,
    chamber=chamber,  # 'upper' or 'lower'
    classification='committee'
)
```

   Notably,

   - the `committee` attribute is now `name`
   - if you were manually accessing `committee['members']` for any reason, that information is now in `_related`; a common use of this is checking whether any members were saved: like here

   If you're instantiating a subcommittee, you'll need to pass `parent_id` as an argument as well. `parent_id` can be:

   - another instance of `Organization`
   - a dictionary, like `{'name': 'Appropriations', 'classification': 'lower'}` which will find the House Appropriations committee at import-time

   Example diff: NC committees conversion

5. Running your committee scraper:

   Just like before, we should now be able to run our scraper:

```
$ docker-compose run scrape nc committees
```

   Among the output should be:

```
nc (scrape)
  committees: {}
committees scrape:
  duration:  0:00:06.125869
  objects:
    membership: 1114
    organization: 59
```

Check that this organization count is in line with what you expect from the state.

And at the end the billy output, the number of committees should be the same:

```
02:42:17 INFO billy: imported 59 committee files
```

At this point, your committee scraper is ready to go. Be sure to commit, and update your PR!

Converting Bills

Bill scrapers are more complex, but conversion to pupa still follows the same basic principles.

1. Update the imports and class definition:

```
# old
from billy.scrape.bills import BillScraper, Bill

# new
from pupa.scrape import Scraper, Bill
```

and:

```
# old
class NCBillScraper(BillScraper):
    jurisdiction = 'nc'

# new
class NCBillScraper(Scraper):
```

2. Using the same pattern as you did with people and committees, add the bill scraper's class name to the state's metadata file.

3. Update the `scrape` method:

The billy scrape method was `scrape(session, chambers)`, and required both parameters.

We can change it to look something like:

```
def scrape(self, session=None, chamber=None):
    if not session:
        session = self.latest_session()
        self.info('no session specified, using %s', session)

    chambers = [chamber] if chamber else ['upper', 'lower']
    for chamber in chambers:
        yield from self.scrape_chamber(chamber, session)
```

4. Update the usage of `Bill` and its methods:

There are a lot of small changes from billy to pupa regarding bills; here are examples of the main alterations needed:

Altering the names of the constructor's arguments:

```
# old
Bill(session, chamber, bill_id, title, type=bill_type)

# new
Bill(
```

```
    bill_id,
    legislative_session=session,  # A session name from the metadata's␣
↪`legislative_sessions`
    chamber=chamber,  # 'upper' or 'lower'
    title=title,
    classification=bill_type  # eg, 'bill', 'resolution', 'joint␣
↪resolution', etc.
)
```

Adding versions and documents:

```
# old
bill.add_version(name, url, mimetype='text/html')

# new
bill.add_version_link(
    note,  # Description of the version from the state; eg, 'As introduced
↪', 'Amended', etc.
    url,
    media_type='text/html'  # Still a MIME type
)

# And analogous for documents, using `add_document_link()`
```

Note: If there is an `on_duplicate` parameter, most likely you'll want to replace it with `on_duplicate='ignore'`, but it may be worth discussion on the Open States Slack or on the pupa GitHub ticket.

Adding sponsors:

```
# old
bill.add_sponsor(type, name, chamber=chamber)

# new
bill.add_sponsorship(
    name,  # Sponsor's name
    classification=spon_type,  # The state's classification; eg, 'co-
↪sponsor', 'co-author', 'primary'
    entity_type='person',  # If the bill is sponsored by a committee,␣
↪this should be 'organization' instead
    primary=is_primary  # Boolean value
)
```

Adding actions:

```
# old
bill.add_action(actor, action, date, type=action_type)

# new
bill.add_action(
    description,  # Action description, from the state
    date,  # `YYYY-MM-DD` format
    chamber=actor,  # 'upper' or 'lower'
    classification=action_type  # Options explained in the next section
)
```

Adding votes:

```
# old
bill.add_vote(vote)

# new - yield vote from scrape
yield vote
```

See "Converting Votes" for details on converting a `Vote` into a `VoteEvent`.

5. Fix action categorization:

   If you try to run the scraper at this point, you'll get an error that the action types fail validation.

   The billy action types have been normalized in Open Civic Data, and the new types are documented there.

   To ease this transition, you can run this utility script to perform an in-place conversion from billy action types to pupa action types.

   To be on the safe side, commit your code prior to running this script, in case it malfunctions unexpectedly.

   ```
   $ ./scripts/convert-actions.sh openstates/nc/bills.py
   ```

   You'll also want to remove any categorization of actions as `'other'`, simply opting for `None` instead.

   At this point, your bill scraper should be ready to go.

   Example diff: NC bill conversion

Converting Votes

Votes are a relatively easy process. There are two major changes:

- The class is now named `VoteEvent` instead of `Vote`.
- Instead of using `'other'` for all votes that aren't a 'yes' or a 'no', types like `'excused'`, `'absent'` and `'not voting'` have been added.

1. Update imports and class definition:

   :: # old from billy.scrape.bills import VoteScraper, Vote

      # new from pupa.scrape import Scraper, VoteEvent

   and:

   ```
   # old
   class NCVoteScraper(VoteScraper):
       jurisdiction = 'nc'

   # new
   class NCVoteScraper(Scraper):
   ```

2. Just like we've done before, add the new class instance to metadata.

3. Update `scrape` method:

   The logic here will be almost identical to what you did in the bill scraper. Note that we need it to scrape the latest session's votes by default.

4. Update usage of `Vote` to `VoteEvent`:

   The old `Vote` constructor took a ton of parameters:

```
# old
Vote(chamber, date, motion, passed,
     yes_count, no_count, other_count, type='other', **kwargs)
```

   Sometimes there'd be even more parameters stuffed into the `kwargs`, like:

```
# also old
Vote(chamber, date, motion, passed,
     yes_count, no_count, other_count, type='other',
     bill_id=bill_id, bill_chamber=bill_chamber, session=session
     )
```

   Be careful, too, since many of the older scrapers pass these parameters in by position alone; it's easy to make mistakes in their order when converting.

   `VoteEvent` requires all parameters to be passed by keyword:

```
# new
VoteEvent(
    chamber=chamber,  # 'upper' or 'lower'
    start_date='2017-03-04', # 'YYYY-MM-DD' format
    motion_text=motion,
    result=passed, # String value, 'pass' or 'fail'
    classification='passage',  # Can also be 'other'

    # Provide a Bill instance to link with the VoteEvent...
    bill=bill_instance,
    # or pass in bill informaion if a Bill instance isn't available.
    legislative_session=bill_session,
    bill=bill_id,
    bill_chamber=bill_chamber
)
```

   Instead of linking a `Bill` to a `VoteEvent` by calling `bill.add_vote(vote)`, a `Bill` instance or identifying bill information is passed directly to the `VoteEvent` constructor.

   You'll notice that in the instantiation of the class we didn't pass `yes_count`, `no_count`, `other_count`. Instead we'll set these using the `set_count` method:

```
vote.set_count('yes', yes_count)
vote.set_count('no', no_count)

# if possible, we'll split 'other' out into more specific values
vote.set_count('absent', absent_count)
vote.set_count('not voting', not_voting_count)
```

   Individual legislators's votes are added to the `VoteEvent` in the same way as in billy, with the only exception being `.other`:

```
# these haven't changed between billy and pupa
vote.yes(yes_voter_name)
vote.no(no_voter_name)

# old
vote.other(other_voter_name)
```

```
# new
vote.vote('not voting', not_voting_name)
vote.vote('absent', absentee_name)
```

Our example state of NC was a bit more complex to change, but nonetheless here's the **example diff:** NC vote conversion

Converting Events

Events are also relatively easy as well as short. :

- The class name is unchanged `Event`.
- We need to localize the time for that we can add a new property _tz to the EventScraper class.

(I will take the example of mi events for clarity, in the meantime it is recommended to have a look at this: MI events diff)

1. Update imports and class definition:

```
# old
from billy.scrape.events import Event, EventScraper

# new
from pupa.scrape import Scraper, Event

and

# old
class MIEventScraper(EventScraper, LXMLMixin):
    jurisdiction = 'mi'
# new
class MIEventScraper(Scraper):
    _tz = pytz.timezone('US/Eastern') # 'US/Eastern' is timezone for mi, you have␣
↪to google the timezone for the state you are working on, available timezones in␣
↪pytz can be seen with "pytz.all_timezones".
```

2. Just like we've done before, add the new class instance to metadata.
3. Update `scrape` method:

   The logic here will be almost identical to what you did in the bill,vote scraper. Note that we need it to scrape the latest session's votes by default. `yield event` instead of using `self.save_event(event)`.

4. Update usage of `Event`:

   The old `Event` constructor took a ton of parameters:

```
# old
Event(session, datetime, 'committee:meeting', title, location=where)
```

   Be careful, too, since many of the older scrapers pass these parameters in by position alone; it's easy to make mistakes in their order when converting.

   The new `Event` requires all parameters to be passed by keyword:

```
# new
event = Event(
```

```
        name=title,
        start_time=self._tz.localize(datetime),
        timezone=self._tz.zone,
        location_name=where,
        )
```

`add_source` will not change for the new `Event`.

Adding participants have not changed much:

```
# old
event.add_participant('chair', chair_name, 'legislator', chamber=chamber)

# new
event.add_participant(chair_name, type='legislator', note='chair') # Here␣
↪type can be anything "legislator"/"committee" etc, note can also be
↪"chair"/"host" etc.
```

Adding related Bill to the event:

```
# old
event.add_related_bill(
        bill_id=related_bill,
        type='consideration',
        description=relation
)

# new
item = event.add_agenda_item(relation)
item.add_bill(related_bill)
```

Adding agenda to the event, previously were passing `agenda` as a parameter directly to event at definition time:

```
# old
Event(..,agenda=agenda,..)
# new
event.add_agenda_item(agenda)
```

Ensuring code quality

Along with the conversion from billy to pupa, we're also using the flake8 utility to ensure code quality. To check for code quality warnings, add your state directory to the *flake8.sh* script. Then run *./flake8.sh* and fix any warnings it reports.

CHAPTER 3

Open States Infrastructure

Open States has grown to be a rather complex project over the years. The purpose of this documentation is to help explain how Open States works end-to-end to better help contributors and project members grasp the entirety of the system.

Overview

A great deal of Open States' infrastructure falls within the billy project. Here's roughly how it works:

- A scraper is written that utilizes `billy.scrape`'s Python helpers.
- When invoked via `billy-update`, this scraper writes JSON files to disk.
- These JSON files are then imported by `billy.importers` into MongoDB.
- A denormalization step takes place allowing us to have aggregate info, `billy.reports`.

At this point the data has been scraped, validated, post-processed, imported, and we've generated some statistics like how many bills were updated, etc.

From here, the data is served out via a Django project. `billy` also helps with this, in the form of three applications:

`billy.web.public` Essentially the site you see when you browse OpenStates.org.

`billy.web.api` Open States API v1

`billy.web.admin` This is a custom-built admin, truthfully just a series of views that project maintainers have found useful over time. There are some tools for manual data entry/reconciliation as well as error reporting.

Note: Now that the project is more community-driven and doesn't have a full-time staff, this approach is somewhat outdated and we'll be looking to improve access so that non-staff can help with some of the tasks included in the admin.

CHAPTER 4

Policies

Standardizing legislative information across states is a hairy task and sometimes we have to make compromises. We'll do our best to both articulate the policy and (where necessary) the rationale.

Categorization

One of the ways that we add value to the data we provide is by attempting to categorize bills, actions, and votes across states.

Bill Types

State legislatures deal with more than bills. Despite the name of the bill objects in our data we take in all types of legislation that a state might produce. Generally looking at the bill_id will help you determine the type of legislation, but to make things easier across states we provide a *type* field on bills. This field is a list with one (or more) of the following values:

Common Values: * bill * resolution * joint resolution * concurrent resolution * constitutional amendment

Some states also make use of additional types such as 'contract', 'nomination', 'memorial' and more.

Action Types

Although most states follow very similar parlimentary procedure the names that their bill status systems use for various actions almost never match up. To make analysis and the building of certain types of tools easier we attempt to categorize about 30 types of common actions. In using our data you'll find these values in the *type* field of actions.

- **bill:introduced** - Bill is introduced or prefiled
- **bill:passed** - Bill has passed a chamber
- **bill:failed** - Bill has failed to pass a chamber
- **bill:withdrawn** - Bill has been withdrawn from consideration

- **bill:veto_override:passed** - The chamber attempted a veto override and succeeded
- **bill:veto_override:failed** - The chamber attempted a veto override and failed
- **bill:reading:1** - A bill has undergone its first reading
- **bill:reading:2** - A bill has undergone its second reading
- **bill:reading:3** - A bill has undergone its third (or final) reading
- **bill:filed** - A bill has been filed (for states where this is a separate event from bill:introduced)
- **bill:substituted** - A bill has been replaced with a substituted wholesale (called hoghousing in some states)
- **governor:received** - The bill has been transmitted to the governor for consideration
- **governor:signed** - The bill has signed into law by the governor
- **governor:vetoed** - The bill has been vetoed by the governor
- **governor:vetoed:line-item** - The governor has issued a line-item (partial) veto
- **amendment:introduced** - An amendment has been offered on the bill
- **amendment:passed** - The bill has been amended
- **amendment:failed** - An offered amendment has failed
- **amendment:amended** - An offered amendment has been amended (seen in Texas)
- **amendment:withdrawn** - An offered amendment has been withdrawn
- **amendment:tabled** - An amendment has been 'laid on the table' (generally preventing further consideration)
- **committee:referred** - The bill has been referred to a committee
- **committee:passed** - The bill has been passed out of a committee
- **committee:passed:favorable** - The bill has been passed out of a committee with a favorable report
- **committee:passed:unfavorable** - The bill has been passed out of a committee with an unfavorable report
- **committee:failed** - The bill has failed to make it out of committee
- **other** - All other actions will have a type of "other"

Vote Types

Similarly to actions, we make an effort to categorize the motion being voted upon. You'll find these values in the *type* field of vote objects in billy or the *categorization* field in pupa.

Possible values:

- **passage** - This is a vote to pass (either out of committee or a chamber)
- **amendment** - Vote on amending a bill
- **veto_override** - Vote to override an executive veto
- **reading:1** - Vote on a first reading
- **reading:2** - Vote on a second reading
- **other** - All other votes

Subjects

Many states provide a list of subject areas for individual pieces of legislation. We've made an attempt to map these to a comprehensive set of subjects.

If you're using the API data you'll find these in the *subjects* field if we've been able to categorize a state's bills. If you're interested in the state's native categories those can be found in *scraped_subjects* field.

- Agriculture and Food
- Animal Rights and Wildlife Issues
- Arts and Humanities
- Budget, Spending, and Taxes
- Business and Consumers
- Campaign Finance and Election Issues
- Civil Liberties and Civil Rights
- Commerce
- Crime
- Drugs
- Education
- Energy
- Environmental
- Executive Branch
- Family and Children Issues
- Federal, State, and Local Relations
- Gambling and Gaming
- Government Reform
- Guns
- Health
- Housing and Property
- Immigration
- Indigenous Peoples
- Insurance
- Judiciary
- Labor and Employment
- Legal Issues
- Legislative Affairs
- Military
- Municipal and County Issues
- Nominations

- Other
- Public Services
- Recreation
- Reproductive Issues
- Resolutions
- Science and Medical Research
- Senior Issues
- Sexual Orientation and Gender Issues
- Social Issues
- State Agencies
- Technology and Communication
- Trade
- Transportation
- Welfare and Poverty

Session Naming

States name their sessions drastically differently, and sometimes inconsistently even within their own site. (49th vs 2008 Regular Session). As our goal is to help smooth these inconsistencies we put forward this guide to naming sessions within state metadata. (See https://github.com/sunlightlabs/openstates/issues/81 for discussion on the topic)

Default Session Names

The `sessions` list within `terms` is dangerous to change as all bill data is keyed off it. As a rule these should be short and generally useful for the scraper to make the appropriate decisions on what data to scrape.

If a state calls its 1st special session in 2010 '2010E1' this is a perfectly acceptable name for the session in the metadata. Similarly 49th-regular, 2009-Special-B, etc. are fine names. Generally names with spaces should be avoided simply for ease of construction of URLs, etc. In states where spaces are already in use it is fine to continue to use them.

The one caveat is that if a state uses a unique ID that has no bearing on the session itself such as '7323' for the 2011 session, this *should not* be used. Instead add some mapping that maps a session name that is descriptive to their internal ids.

Session Display Names

Because the most convenient name to refer to a session is often far from what a user might expect to see upon opening a mobile application, the `session_details` dict supports a `display_name` key.

Suitable display names are descriptive but also short and obey a given style.

General Rules

- All sessions should be in title case.
- Fewer than 20 characters is highly preferable.
- Months should be abbreviated to 3 letters (Jan., Feb., Jun., Dec.)

Ordinals

If no special sessions are present:

- [Ordinal] Legislature

If special sessions are present:

- [Ordinal] Regular Session
- [Ordinal], [Ordinal] Special Session

Examples:

- 82nd Legislature
- 82nd Regular Session
- 82nd, 3rd Special Session

Years

- [Year/Year-Range] Regular Session
- [Year/Year-Range], [Ordinal] Special Session
- [Mon. Year] Special Session

Examples:

- 2010 Regular Session
- 2011-2012, 4th Special Session
- Dec. 2011 Special Session

State API Keys

Unfortunately, some states find it necessary to require API Keys (or other credentials) to access their best data.

Despite the difficulties this creates for contributors, in the interest of ensuring we have the best possible data we've made the decision that we will use this data where possible.

Our policy:

- We will maintain (when possible) two copies of credentials, one for development and one for production. (Thus minimizing the chance that a mistake made w/ a development key will jeopardize our ability to scrape.)
- We encourage developers to get an API key of their own, but if necessary we can share our testing key in limited circumstances.

Currently only a few states require API keys:

- New York - http://legislation.nysenate.gov/static/docs/html/index.html

  - Request Form: http://legislation.nysenate.gov/

- Indiana - http://docs.api.iga.in.gov/api.html
  - API Key Request Process: Email Bob Amos (bob.amos@iga.in.gov or bamos@iga.in.gov), and include your name, address, phone, email address and company. Also indicate that you have read the terms of service at the link above.
- Oregon - https://www.oregonlegislature.gov/citizen_engagement/Pages/data.aspx
  - API Credentials Request Process: Email help.leg@oregonlegislature.gov and include your name, e-mail address, company or organization name, and contact phone number. Also please read over the API agreement and let them know you agree to the terms in the email.

If you're in need of an API key and unable to via these channels please contact a member of the core team to discuss getting access to the development key.

Testing Scrapers

One of the first things people new to the project tend to notice is that there aren't a lot of tests in the scrapers.

Over the years we've evolved a de facto policy of somewhat discouraging tests, which is definitely an unusual stance to take and warrants explanation.

Intentionally Fragile Scrapers

When it comes to scrapers, there are two major types of breakage:

1. the scraper collects bad information and inserts it into the database
2. the scraper encounters an error and quits without importing data

Given a choice, the second is greatly preferable. Once bad data makes it into the database, it can be difficult to detect and remove. On the other hand, the second can be triggered to alert us immediately and someone can evaluate the proper fix.

The best way to favor the second over first is to write "intentionally fragile" scrapers. That is, scrapers that raise an exception when they see unexpected input.

While it is possible to try to write a resilient scraper that recovers, by nature these scrapers are more likely to produce the first kind of error, and so we encourage scraper writers to be conservative in what errors are suppressed.

Here's an example of an overly permissive scraper:

```
party_abbr = doc.xpath('//span[@class="partyabbr"])
if party_abbr == 'D':
    party = 'Democratic'
elif party_abbr == 'R':
    party = 'Republican'
else:
    # haven't seen this yet, but let's just keep things moving
    party = party_abbr
```

The following would be preferred:

```
party_abbr = doc.xpath('//span[@class="partyabbr"])
party = {'D': 'Democratic', 'R': 'Republican'}[party_abbr]
```

This code would raise a `KeyError` the first time a new party is found. This forces someone to take a look, fix the scraper with an entry for the new party, and then the scraper will be able to run again with correct data.

Testing Scrapers Is Hard

On most software projects a failing test means that something is broken, and passing tests should mean that things are working just fine.

In our experience however, the majority of the "breaks" that occur in scrapers are due to upstream site changes.

In the past the fragile nature of scrapers has led to people writing a lot of bad tests, which is where our stance of somewhat discouraging tests has come from. An example of a bad test:

```
def extract_name(doc):
    return doc.xpath('//h2[@class="legislatorName"]').text_content().strip()


def test_extract_name():
    # probably a snapshot of the page at some point in time
    EXAMPLE_LEGISLATOR_HTML = '...'

    doc = lxml.html.fromstring(EXAMPLE_LEGISLATOR_HTML)
    assert extract_name(doc) == 'Erica Example'
```

With a test like this:

- As soon as the HTML changes, the scraper will start failing, but the tests will still pass.
- The scraper will then be updated, breaking the test.
- The test HTML will be updated, fixing the test.

But since the initial scraper breakage isn't predicted by a failing test, this type of test really doesn't serve us any purpose and just results in extra code to maintain every time the scraper needs a slight change.

Other Strategies

Of course this isn't to say that we just abandon the idea of testing, altogether.

If you're more comfortable writing tests, say you're parsing a particularly nasty PDF and want to run it against some test data: a test might make sense there as a way to be confident in your own code, by all means, write a test.

We also have some other strategies to help ensure data quality:

Validate Scraper Output

Scraper output is verified against JSON schemas that protect against common regressions (missing sources, invalid formatted districts, etc.) - most of these tests can be written effectively against scraper output across the board, and in doing so also applies universally across all 50 states.

We also aim for our underlying libraries like billy to be as well tested as possible. (To be 100% clear, our lax testing philosophy only applies to site-specific scraper code, not these support libraries.)

Run Scrapers Regularly

In a sense, the scrapers are tested every night by being run. This is why the intentionally fragile approach is so important; those failures are in essence the same as integration test failures. Of course, this doesn't tell us if the scraper is picking up bad data, etc., but combined with validation we can be fairly confident in our data.

Test Utilities

One area we can definitely improve upon is our use of (and then thorough testing of) common functions. Right now (largely because of the great variety of authors, etc.) many scrapers do similar things like conversion of party abbreviations and whitespace normalization in slightly different ways. We should be making a push to use common utility functions and thoroughly test those.

CHAPTER 5

Related Projects

billy Python backend for Open States' scrapers and website.

pyopenstates Python client library for Open States API.