

Seeding MongoDB Database

Motivation for Database Seeding

- Pre-populating the database can enhance developer productivity
- It facilitates simple exploring of various scenarios
- Is particularly useful in establishing normalised data models with inter-document relationships
- Can also be used to pre-configure database for production



Mongoose seeders on NPM

- Variety of modules available
- Most fairly simple



mongoose-seed public



Seed data population for Mongoose

mongoose-seed lets you populate and clear MongoDB documents with all the benefits of Mongoose validation

mongoose-seed-plus public



Seed data population for Mongoose

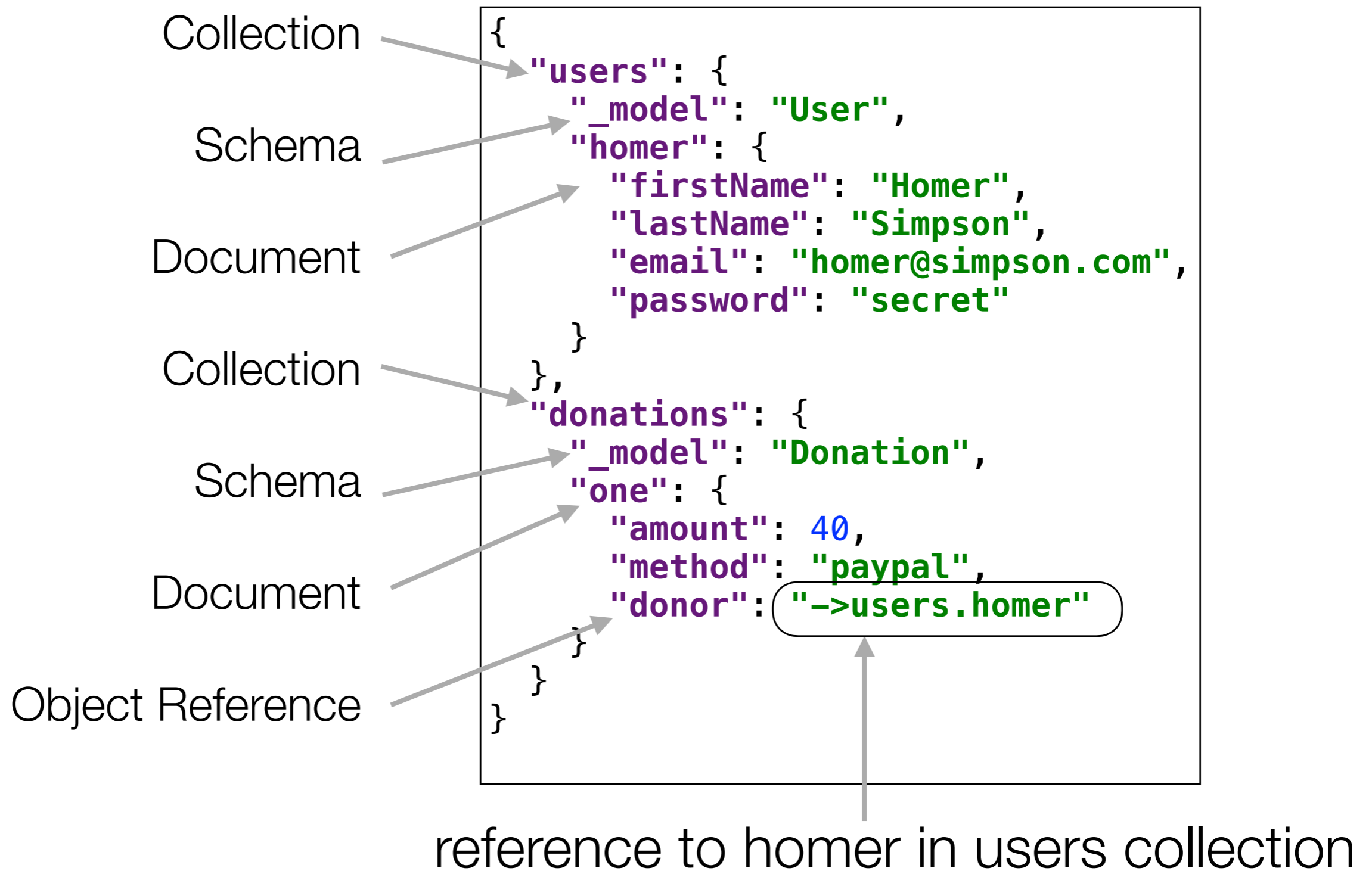
mongoose-seed-plus lets you populate and clear MongoDB documents with all the benefits of Mongoose validation

<https://github.com/SamVerschueren/mongoose-seeder>

- mongoose-seeder loads from an enhanced JSON file
- Includes special notation for loading relationships between documents

The screenshot shows the GitHub repository page for `mongoose-seeder` by `SamVerschueren`. The repository has 55 commits, 1 branch, 6 releases, and 1 contributor. The latest commit is `d692376` on 15 Sep 2015. The repository includes a `test` directory and several files: `.gitignore`, `.jshintrc`, `.travis.yml`, `CHANGELOG.md`, `LICENSE`, `README.md`, `index.js`, and `package.json`. The `README.md` file is expanded, showing the title `mongoose-seeder` and two badges: `build passing` and `coverage 99%`. The text in the `README.md` describes the library's purpose: "When testing an application, you always want to start with the same database. It is a lot of work to manually create dummy data and link them together. When you want extra data to test with, you'll have to create your mongoose objects manually in the `before` method of the entire testsuite. This library offers a nice, clean and elegant solution that will create the dummy data objects from a JSON file."

initdata.json



- On application startup:

db.js

- Delete any existing data in the collections
- Populate the database on initial connection during startup
- Only if NOT in 'production' mode

```
mongoose.connection.on('connected', function () {
  console.log('Mongoose connected to ' + dbURI);
  if (process.env.NODE_ENV !== 'production') {
    const seeder = require('mongoose-seeder');
    const data = require('./initdata.json');
    const Donation = require('./donation');
    const User = require('./user');
    seeder.seed(data, { dropDatabase: false, dropCollections: true }).then(dbData => {
      console.log('preloading Test Data');
      console.log(dbData);
    }).catch(err => {
      console.log(error);
    });
  }
});
```

db.js

```
const seeder = require('mongoose-seeder');  
const data = require('./initdata.json');  
const Donation = require('./donation');  
const User = require('./user');
```

- import
 - the seeder library
 - the initial data json
 - the Schemas
 - Donation
 - User

db.js

- Call the seed component
 - pass 'data' loaded from JSON file
 - options:
 - keep the database
 - delete contents of all collections

```
const seeder = require('mongoose-seeder');
const data = require('./initdata.json');
const Donation = require('./donation');
const User = require('./user');
seeder.seed(data, { dropDatabase: false, dropCollections: true })
```


db.js

- Promise success:
 - Log loaded data to console
- Promise fail:
 - Log error message


```
const seeder = require('mongoose-seeder');
const data = require('./initdata.json');
const Donation = require('./donation');
const User = require('./user');
seeder.seed(data, { dropDatabase: false, dropCollections: true }).then(dbData => {
  console.log('preloading Test Data');
  console.log(dbData);
}).catch(err => {
  console.log(error);
});
```

db.js

```
mongoose.connection.on('connected', function () {
  console.log('Mongoose connected to ' + dbURI);
  if (process.env.NODE_ENV !== 'production') {
    const seeder = require('mongoose-seeder');
    const data = require('./initdata.json');
    const Donation = require('./donation');
    const User = require('./user');
    seeder.seed(data, { dropDatabase: false, dropCollections: true }).then(dbData => {
      console.log('preloading Test Data');
      console.log(dbData);
    }).catch(err => {
      console.log(error);
    });
  }
});
```

initdata.json version 2

Donation Donate Report Settings Logout



Amount	Method donated	Donor
40	paypal	Bart Simpson
90	direct	Marge Simpson
430	paypal	Homer Simpson

- We can log in and view basic app functions without having to signup/login and make donations etc...

```
{
  "users": {
    "_model": "User",
    "homer": {
      "firstName": "Homer",
      "lastName": "Simpson",
      "email": "homer@simpson.com",
      "password": "secret"
    },
    "marge": {
      "firstName": "Marge",
      "lastName": "Simpson",
      "email": "marge@simpson.com",
      "password": "secret"
    },
    "bart": {
      "firstName": "Bart",
      "lastName": "Simpson",
      "email": "bart@simpson.com",
      "password": "secret"
    }
  },
  "donations": {
    "_model": "Donation",
    "one": {
      "amount": 40,
      "method": "paypal",
      "donor": "->users.bart"
    },
    "two": {
      "amount": 90,
      "method": "direct",
      "donor": "->users.marge"
    },
    "three": {
      "amount": 430,
      "method": "paypal",
      "donor": "->users.homer"
    }
  }
}
```