Database interactions in Express

Completing the express local library app to include create, retrieve edit and delete.

Copy the folder express_app4 to a new folder express_app5 and change directory to this in two open console windows.

Genre details

So far documents have been retrieved for full collections, now the query must be refined to allow searching for one item by id.

The next step should use the route in catalog.js:

```
// GET request for one Genre.
router.get('/genre/:id', genre_controller.genre_detail);
```

This should call the controller genre_detail to check that the genre id is correct and to return a list of all books with that genre id.

Open controllers/genreController.js and add the code to import the Book and async modules at the top.

```
var Genre = require('../models/genre');
var Book = require('../models/book');
var async = require('async');
```

Replace

```
// Display detail page for a specific Genre.
exports.genre_detail = function(req, res) {
    res.send('NOT IMPLEMENTED: Genre detail: ' + req.params.id);
};
```

with:

```
// Display detail page for a specific Genre.
exports.genre_detail = function(req, res, next) {

    async.parallel({
        genre: function(callback) {
            Genre.findById(req.params.id)
              .exec(callback);
        },

        genre_books: function(callback) {
            Book.find({ 'genre': req.params.id })
              .exec(callback);
        },
    }, function(err, results) {
        if (err) { return next(err); }
        if (results.genre==null) { // No results.
            var err = new Error('Genre not found');
            err.status = 404;
            return next(err);
        }
        // Successful, so render
        res.render('genre_detail', { title: 'Genre Detail', genre: 
results.genre, genre_books: results.genre_books } );
    });

};
```

The controller makes two queries using async. The first looks to find the genre corresponding to the id, thereby checking it exists. If this does not exist we will report an error status 404 and message ‘Genre not found’.

The second query searches for books whose genres matches the genre id.

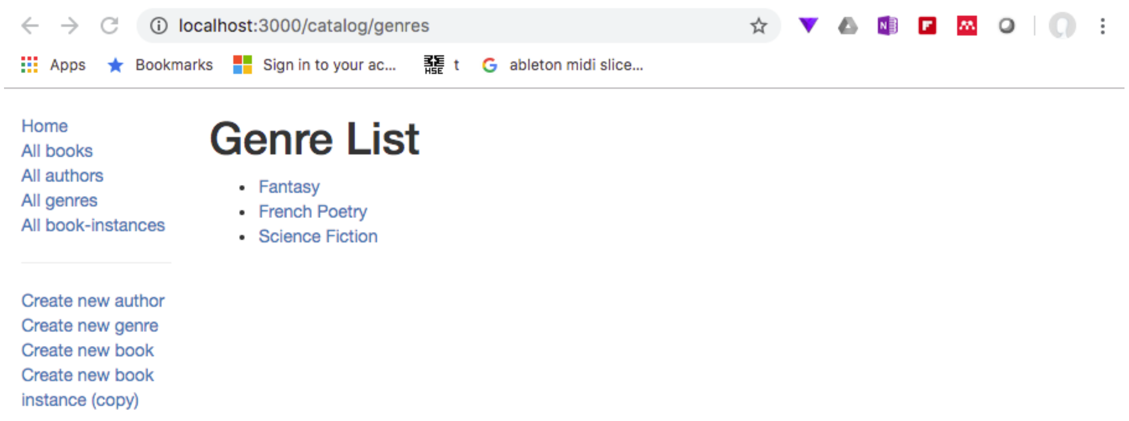
Only if both queries are succesful will a list of titles be printed.

Create views/genre_detail.pug to display this list with content:

```
extends layout

block content

  h1 Genre: #{genre.name}

  div(style='margin-left:20px;margin-top:20px')

    h4 Books

    dl
      each book in genre_books
        dt
          a(href=book.url) #{book.title}
        dd #{book.summary}

      else
        p This genre has no books
```

Follow the all genres link from the catalog page or

http://localhost:3000/catalog/genres


localhost:3000/catalog/genres

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance (copy)

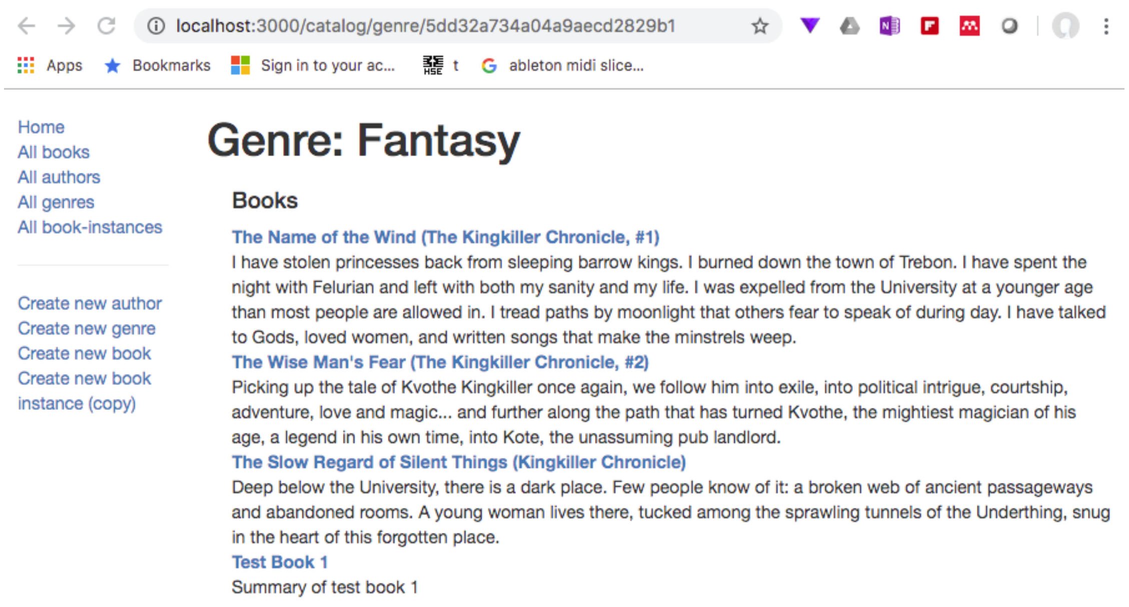
Genre List

- Fantasy
- French Poetry
- Science Fiction

Genre List

click on the Fantasy list or

http://localhost:3000/catalog/genre/5dd32a734a04a9aecd2829b1


localhost:3000/catalog/genre/5dd32a734a04a9aecd2829b1

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance (copy)

Genre: Fantasy

Books

The Name of the Wind (The Kingkiller Chronicle, #1)
I have stolen princesses back from sleeping barrow kings. I burned down the town of Trebon. I have spent the night with Felurian and left with both my sanity and my life. I was expelled from the University at a younger age than most people are allowed in. I tread paths by moonlight that others fear to speak of during day. I have talked to Gods, loved women, and written songs that make the minstrels weep.
The Wise Man's Fear (The Kingkiller Chronicle, #2)
Picking up the tale of Kvothe Kingkiller once again, we follow him into exile, into political intrigue, courtship, adventure, love and magic... and further along the path that has turned Kvothe, the mightiest magician of his age, a legend in his own time, into Kote, the unassuming pub landlord.
The Slow Regard of Silent Things (Kingkiller Chronicle)
Deep below the University, there is a dark place. Few people know of it: a broken web of ancient passageways and abandoned rooms. A young woman lives there, tucked among the sprawling tunnels of the Underthing, snug in the heart of this forgotten place.
Test Book 1
Summary of test book 1

Fantasy Books

Book details

The next query should use the route in catalog.js

```
// GET request for one Book.
router.get('/book/:id', book_controller.book_detail);
```

In the file controllers/bookController.js replace:

```
// Display detail page for a specific book.
exports.book_detail = function(req, res) {
    res.send('NOT IMPLEMENTED: Book detail: ' + req.params.id);
};
```

by the code:

```
// Display detail page for a specific book.
exports.book_detail = function(req, res, next) {

    async.parallel({
        book: function(callback) {

            Book.findById(req.params.id)
              .populate('author')
              .populate('genre')
              .exec(callback);
        },
        book_instance: function(callback) {

          BookInstance.find({ 'book': req.params.id })
          .exec(callback);
        },
    }, function(err, results) {
        if (err) { return next(err); }
        if (results.book==null) { // No results.
            var err = new Error('Book not found');
            err.status = 404;
            return next(err);
        }
        // Successful, so render.
        res.render('book_detail', { title: results.book.title, book:
results.book, book_instances: results.book_instance } );
    });

};
```

This will asynchronously find a book by id and list its instances.

If the book is not found and error status of 404 is found and the message “Book not found” is raised.

If the book is found and instances exist they are listed. The details of the book are populated from the author and genre tables.

Creat a new file views/book_detail.pug with the contents:

```
extends layout

block content
  h1 Title: #{book.title}
  p #[strong Author:]
    a(href=book.author.url) #{book.author.name}
  p #[strong Summary:] #{book.summary}
  p #[strong ISBN:] #{book.isbn}
  p #[strong Genre:]&nbsp;
    each val, index in book.genre
      a(href=val.url) #{val.name}
      if index < book.genre.length - 1
        |,

  div(style='margin-left:20px;margin-top:20px')
    h4 Copies

    each val in book_instances
      hr
      if val.status=='Available'
        p.text-success #{val.status}
      else if val.status=='Maintenance'
        p.text-danger #{val.status}
      else
        p.text-warning #{val.status}
      p #[strong Imprint:] #{val.imprint}
      if val.status!='Available'
        p #[strong Due back:] #{val.due_back}
      p #[strong Id:]&nbsp;
        a(href=val.url) #{val._id}

    else
      p There are no copies of this book in the library.
```

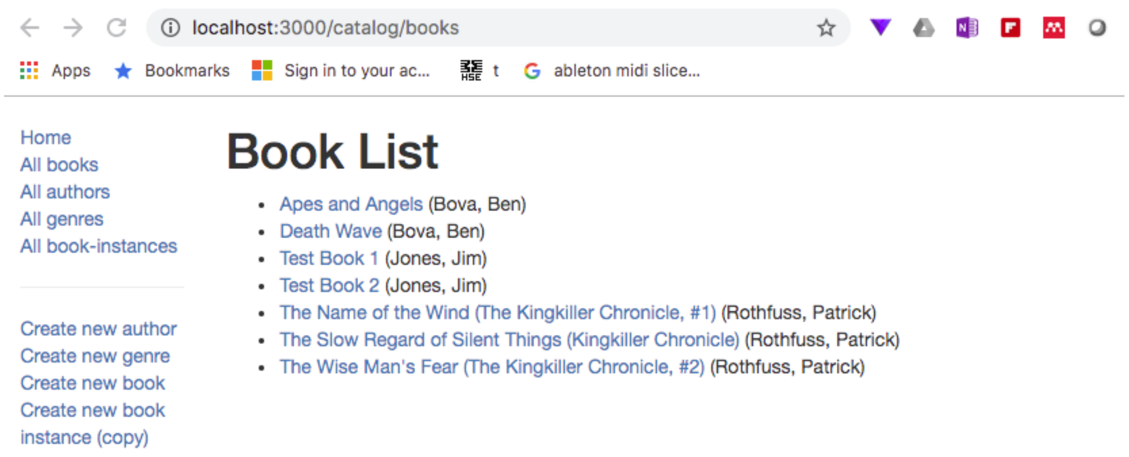
The lines:

```
if index < book.genre.length - 1
        |,
```

add a comma after each instance except the last.

follow the all books link or

http://localhost:3000/catalog/books


localhost:3000/catalog/books

Apps Bookmarks Sign in to your ac... t ableton midi slice...

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance (copy)

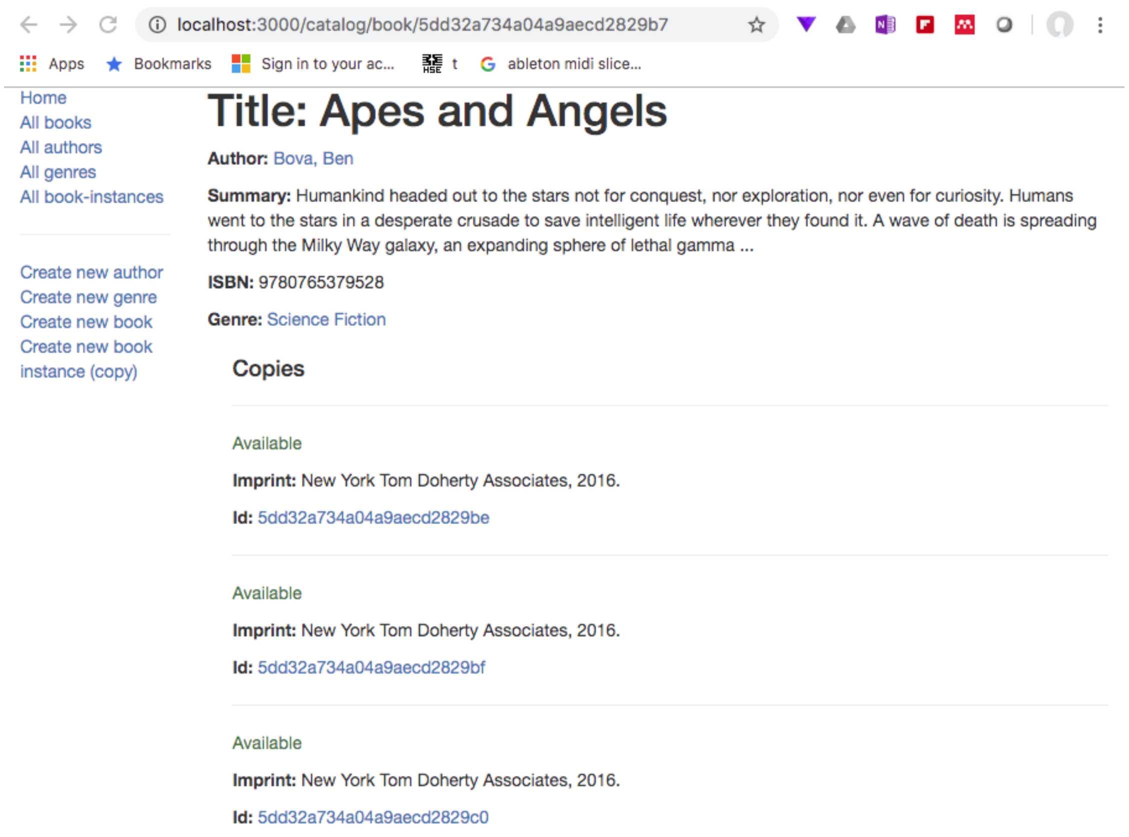
Book List

- Apes and Angels (Bova, Ben)
- Death Wave (Bova, Ben)
- Test Book 1 (Jones, Jim)
- Test Book 2 (Jones, Jim)
- The Name of the Wind (The Kingkiller Chronicle, #1) (Rothfuss, Patrick)
- The Slow Regard of Silent Things (Kingkiller Chronicle) (Rothfuss, Patrick)
- The Wise Man's Fear (The Kingkiller Chronicle, #2) (Rothfuss, Patrick)

book list

Choose Apes and Angels or

localhost:3000/catalog/book/5dd32a734a04a9aecd2829b7


localhost:3000/catalog/book/5dd32a734a04a9aecd2829b7

Apps Bookmarks Sign in to your ac... t ableton midi slice...

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance (copy)

Title: Apes and Angels

Author: Bova, Ben

Summary: Humankind headed out to the stars not for conquest, nor exploration, nor even for curiosity. Humans went to the stars in a desperate crusade to save intelligent life wherever they found it. A wave of death is spreading through the Milky Way galaxy, an expanding sphere of lethal gamma ...

ISBN: 9780765379528

Genre: Science Fiction

Copies

Available

Imprint: New York Tom Doherty Associates, 2016.

Id: 5dd32a734a04a9aecd2829be

Available

Imprint: New York Tom Doherty Associates, 2016.

Id: 5dd32a734a04a9aecd2829bf

Available

Imprint: New York Tom Doherty Associates, 2016.

Id: 5dd32a734a04a9aecd2829c0

book detail

Author detail

The next querey allows listing of all the books by an author identified by id.

Open controllers/authorController.js and add the book model and async module.

```
var Author = require('../models/author');
var Book = require('../models/book');
var async = require('async');
```

Then replace

```
// Display detail page for a specific Author.
exports.author_detail = function(req, res) {
    res.send('NOT IMPLEMENTED: Author detail: ' + req.params.id);
};
```

with code:

```
// Display detail page for a specific Author.
exports.author_detail = function(req, res, next) {

    async.parallel({
        author: function(callback) {
            Author.findById(req.params.id)
              .exec(callback)
        },
        authors_books: function(callback) {
          Book.find({ 'author': req.params.id },'title summary')
          .exec(callback)
        },
    }, function(err, results) {
        if (err) { return next(err); } // Error in API usage.
        if (results.author==null) { // No results.
            var err = new Error('Author not found');
            err.status = 404;
            return next(err);
        }
        // Successful, so render.
        res.render('author_detail', { title: 'Author Detail', author:
results.author, author_books: results.authors_books } );
    });

};
```

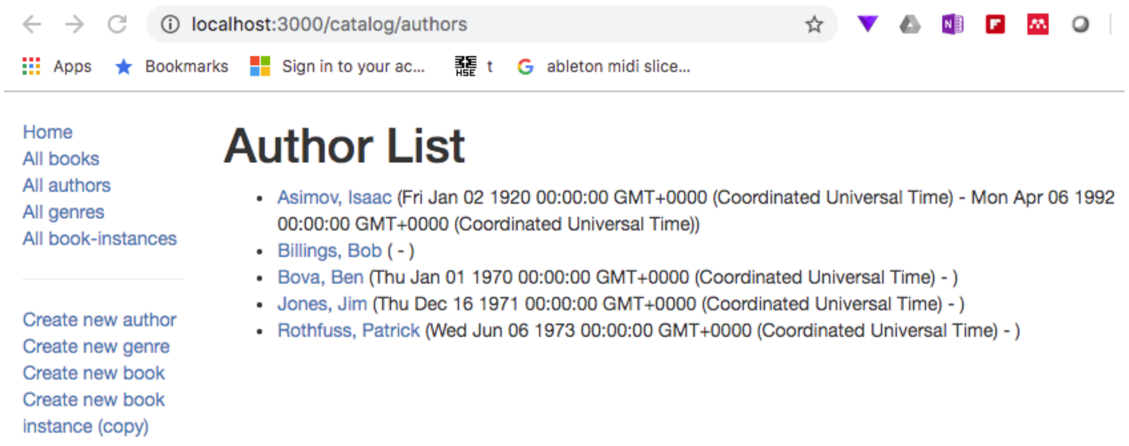
This checkd that the author exists, and sends an error message “Author not found” if it does not.

If authors are foud they are listed by adding a page at views/author_detail.

```
extends layout

block content

  h1 Author: #{author.name}
  p #{author.date_of_birth} - #{author.date_of_death}

  div(style='margin-left:20px;margin-top:20px')

    h4 Books

    dl
      each book in author_books
        dt
          a(href=book.url) #{book.title}
        dd #{book.summary}

      else
        p This author has no books.
```

Follow the all authors link or

http://localhost:3000/catalog/authors



localhost:3000/catalog/authors

Apps · Bookmarks · Sign in to your ac... · t · ableton midi slice...

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance (copy)

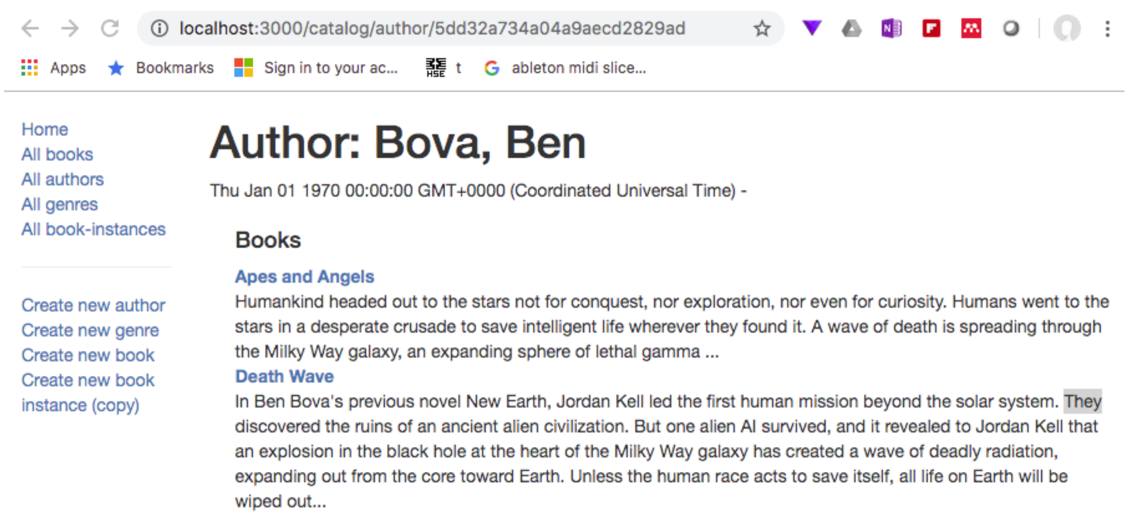
Author List

- Asimov, Isaac (Fri Jan 02 1920 00:00:00 GMT+0000 (Coordinated Universal Time) - Mon Apr 06 1992 00:00:00 GMT+0000 (Coordinated Universal Time))
- Billings, Bob (-)
- Bova, Ben (Thu Jan 01 1970 00:00:00 GMT+0000 (Coordinated Universal Time) -)
- Jones, Jim (Thu Dec 16 1971 00:00:00 GMT+0000 (Coordinated Universal Time) -)
- Rothfuss, Patrick (Wed Jun 06 1973 00:00:00 GMT+0000 (Coordinated Universal Time) -)

author list

Following the link for Ben Bova:

http://localhost:3000/catalog/author/5dd32a734a04a9aecd2829ad



Author: Bova, Ben

Thu Jan 01 1970 00:00:00 GMT+0000 (Coordinated Universal Time) -

Books

Apes and Angels

Humankind headed out to the stars not for conquest, nor exploration, nor even for curiosity. Humans went to the stars in a desperate crusade to save intelligent life wherever they found it. A wave of death is spreading through the Milky Way galaxy, an expanding sphere of lethal gamma ...

Death Wave

In Ben Bova's previous novel New Earth, Jordan Kell led the first human mission beyond the solar system. They discovered the ruins of an ancient alien civilization. But one alien AI survived, and it revealed to Jordan Kell that an explosion in the black hole at the heart of the Milky Way galaxy has created a wave of deadly radiation, expanding out from the core toward Earth. Unless the human race acts to save itself, all life on Earth will be wiped out...

author detail

Bookinstance detail

Open controllers/bookinstanceController.js and replace:

```
// Display detail page for a specific BookInstance.
exports.bookinstance_detail = function(req, res) {
    res.send('NOT IMPLEMENTED: BookInstance detail: ' + req.params.id);
};
```

with code:

```
// Display detail page for a specific BookInstance.
exports.bookinstance_detail = function(req, res, next) {

    BookInstance.findById(req.params.id)
    .populate('book')
    .exec(function (err, bookinstance) {
      if (err) { return next(err); }
      if (bookinstance==null) { // No results.
          var err = new Error('Book copy not found');
          err.status = 404;
          return next(err);
        }
      // Successful, so render.
      res.render('bookinstance_detail', { title: 'Copy: 
'+bookinstance.book.title, bookinstance:  bookinstance});
    })

};
```

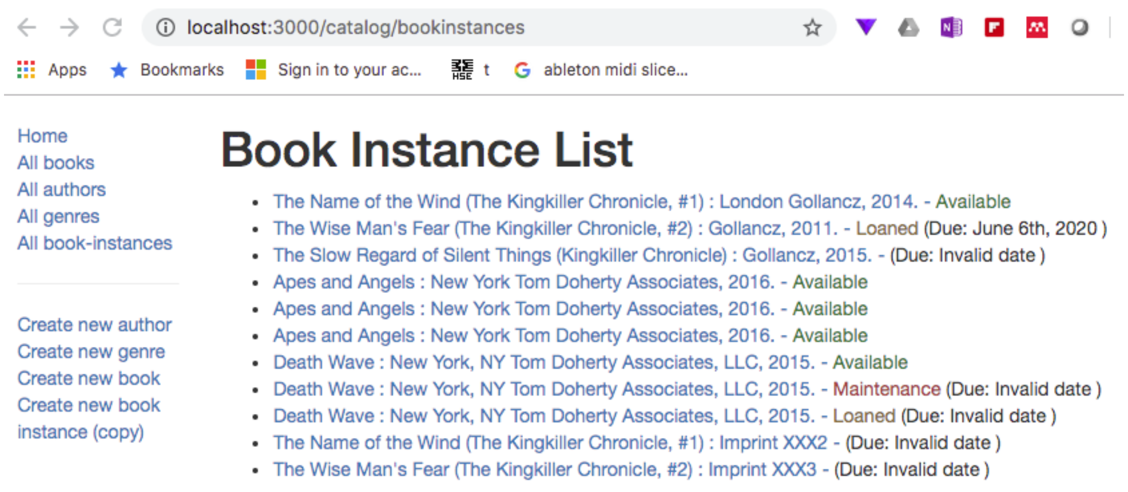
This is a straightforward query which does not need async.

Create a view for this in views/bookinstance_detail.pug

```
extends layout

block content

  h1 ID: #{bookinstance._id}

  p #[strong Title:]
    a(href=bookinstance.book.url) #{bookinstance.book.title}
  p #[strong Imprint:] #{bookinstance.imprint}

  p #[strong Status:]
    if bookinstance.status=='Available'
      span.text-success #{bookinstance.status}
    else if bookinstance.status=='Maintenance'
      span.text-danger #{bookinstance.status}
    else
      span.text-warning #{bookinstance.status}

  if bookinstance.status!='Available'
    p #[strong Due back:] #{bookinstance.due_back}
```

View the 'All book-instances' link or

http://localhost:3000/catalog/bookinstances



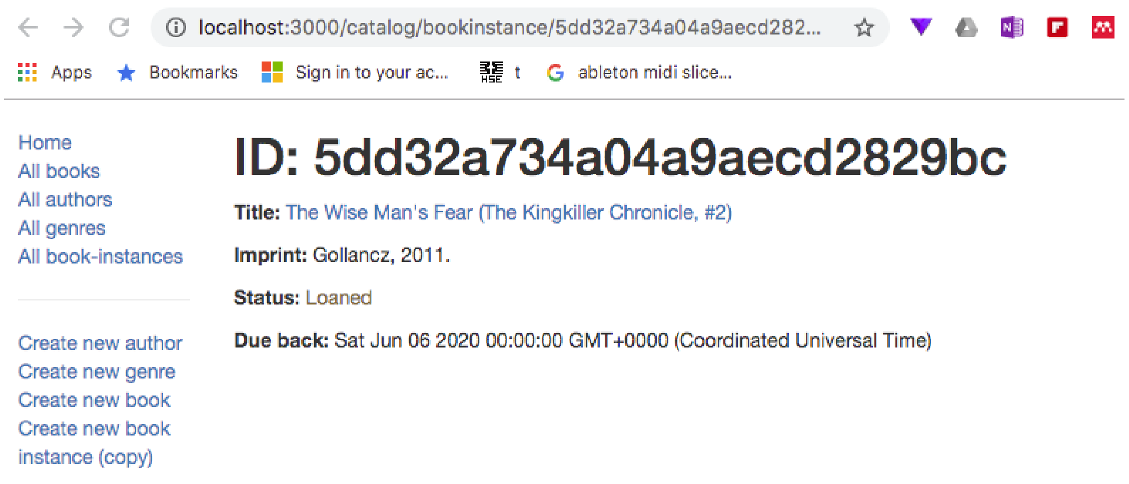
Book Instance List

- The Name of the Wind (The Kingkiller Chronicle, #1) : London Gollancz, 2014. - Available
- The Wise Man's Fear (The Kingkiller Chronicle, #2) : Gollancz, 2011. - Loaned (Due: June 6th, 2020)
- The Slow Regard of Silent Things (Kingkiller Chronicle) : Gollancz, 2015. - (Due: Invalid date)
- Apes and Angels : New York Tom Doherty Associates, 2016. - Available
- Apes and Angels : New York Tom Doherty Associates, 2016. - Available
- Apes and Angels : New York Tom Doherty Associates, 2016. - Available
- Death Wave : New York, NY Tom Doherty Associates, LLC, 2015. - Available
- Death Wave : New York, NY Tom Doherty Associates, LLC, 2015. - Maintenance (Due: Invalid date)
- Death Wave : New York, NY Tom Doherty Associates, LLC, 2015. - Loaned (Due: Invalid date)
- The Name of the Wind (The Kingkiller Chronicle, #1) : Imprint XXX2 - (Due: Invalid date)
- The Wise Man's Fear (The Kingkiller Chronicle, #2) : Imprint XXX3 - (Due: Invalid date)

bookinstance list

Folleo the link to 'The wise mans fear' or

http://localhost:3000/catalog/bookinstance/5dd32a734a04a9aecd2829bc


localhost:3000/catalog/bookinstance/5dd32a734a04a9aecd282...

Apps Bookmarks Sign in to your ac... t ableton midi slice...

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance (copy)

ID: 5dd32a734a04a9aecd2829bc

Title: The Wise Man's Fear (The Kingkiller Chronicle, #2)

Imprint: Gollancz, 2011.

Status: Loaned

Due back: Sat Jun 06 2020 00:00:00 GMT+0000 (Coordinated Universal Time)

bookinstance detail

Tidying up date format

In bookInstance_detail.pug change due_back to due_back formatted.

```
  if bookinstance.status!='Available'

    p #[strong Due back:] #{bookinstance.due_back_formatted}
//-    p #[strong Due back:] #{bookinstance.due_back}
```

Challenge: Update the Author module to add a lifespan virtual property. The lifespan should look like: date_of_birth - date_of_death, where both values have the same date format as BookInstance.due_back_formatted.

Use Author.lifespan in all views where you currently explicitly use date_of_birth and date_of_death.

Modifying data using HTML forms

The browser will request a page containing a form. The web server will create an and return a page with an empty form.

The user populates the form with date. The app then validates this data. If the data is invalid a new form is generated with all the correct user generated content. If the data is valid appropriate actions are carrid out using the valid data (the database is modified), the browser is redefined to a 'success message'.

The blank form will be sent in response to an HTTP GET request and the data will be validated (and processed) in response to an HTTP POST request.

Express needs to use middleware to process POST and GET parameters from forms and to validate values.

Validation checks that values for each field match the required range and format.

Sanitization removes or replaces characters which might represent malicious content.

Express validator will be used to validate and sanitize the forms. Add this to package.json. This will require a rebuild to incorporate it into the image.

```
{
  "name": "myapp",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "nodemon -L start.js"
  },
  "dependencies": {
    "cookie-parser": "~1.4.4",
    "debug": "~2.6.9",
    "express": "~4.16.1",
    "http-errors": "~1.6.3",
    "morgan": "~1.9.1",
    "pug": "2.0.0-beta11",
    "nodemon": "1.19.1",
    "mongoose":"^5.7.8",
    "body-parser":"^1.19.0",
    "async":"3.1.0",
    "moment":"2.24.0",
    "express-validator":"^6.2.0"
  },
  "nodemonConfig": {
    "delay": "1500",
    "verbose": "true"
  }
}
```

```
    docker-compose -f stack.yml down
    docker build -t dt/express-development .
```

```
...
Successfully built 6d142d1ebbfc
```

Instructions for express-validator are found on github.

A quick summary of relevant validation is included in the tutorial.

To maintain a simpler approach to form design two rules will be applied.

1. You can only create new objects if their component objects (such as author and genre) already exist.

2. You can only delete an object (e.g. Book) if it is not referenced by any other objects (e.g. bookInstances).

Routes will be formed in pairs for identical requests to GET and POST.

Create Genre form

Require the express validator at the top of the controllers/genreControllers.js file.

```
var Genre = require('../models/genre');
var Book = require('../models/book');
var async = require('async');
const validator = require('express-validator');
```

in controllers/genreControllers.js replace:

```
// Display Genre create form on GET.
exports.genre_create_get = function(req, res) {
    res.send('NOT IMPLEMENTED: Genre create GET');
};
```

with code:

```
// Display Genre create form on GET.
exports.genre_create_get = function(req, res, next) {
  res.render('genre_form', { title: 'Create Genre' });
};
```

This will render the genre_form pug view passing a variable for the title.

Then replace the controller section for genre_create_post:

```
// Handle Genre create on POST.
exports.genre_create_post = function(req, res) {
    res.send('NOT IMPLEMENTED: Genre create POST');
};
```

with:

```
// Handle Genre create on POST.
exports.genre_create_post =  [

  // Validate that the name field is not empty.
  validator.body('name', 'Genre name required').isLength({ min: 1 }).trim(),

  // Sanitize (escape) the name field.
  validator.sanitizeBody('name').escape(),

  // Process request after validation and sanitization.
  (req, res, next) => {

    // Extract the validation errors from a request.
    const errors = validator.validationResult(req);

    // Create a genre object with escaped and trimmed data.
    var genre = new Genre(
      { name: req.body.name }
    );


    if (!errors.isEmpty()) {
      // There are errors. Render the form again with sanitized values/error
messages.
      res.render('genre_form', { title: 'Create Genre', genre: genre, errors:
errors.array()});
      return;
    }
    else {
      // Data from form is valid.
      // Check if Genre with same name already exists.
      Genre.findOne({ 'name': req.body.name })
        .exec( function(err, found_genre) {
          if (err) { return next(err); }

          if (found_genre) {
            // Genre exists, redirect to its detail page.
            res.redirect(found_genre.url);
          }
          else {

            genre.save(function (err) {
              if (err) { return next(err); }
              // Genre saved. Redirect to genre detail page.
              res.redirect(genre.url);
            });

          }

        });
    }
  }
];
```

This is not a single function but an array of functions.

Validator checks that the name field is not empty and trims any blank space.

Sanitize removes any dangerous characters from the field.

Processing the request uses isEmpty to check for errors from validation and sanitization

If there are errors then render the empty page again.

If no errors then check if genre already exists. If it does, go to the details page.

If genre does not exist it is saved and then the details page is displayed.

This will be a general pattern.

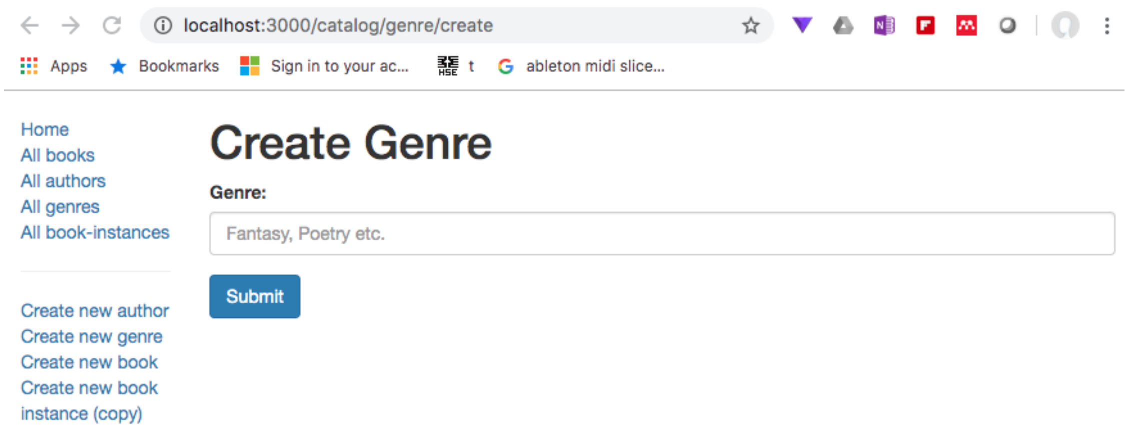
A single view is created in views/genre_form.pug the GET query will lead to a title and and empty form, in the POST query the validated and sanitized data is displayed.

```
extends layout

block content
  h1 #{title}

  form(method='POST' action='')
    div.form-group
      label(for='name') Genre:
      input#name.form-control(type='text', placeholder='Fantasy, Poetry etc.'
name='name' value=(undefined===genre ? '' : genre.name))
    button.btn.btn-primary(type='submit') Submit

  if errors
   ul
    for error in errors
     li!= error.msg
```

The form has a single field of type text called name.

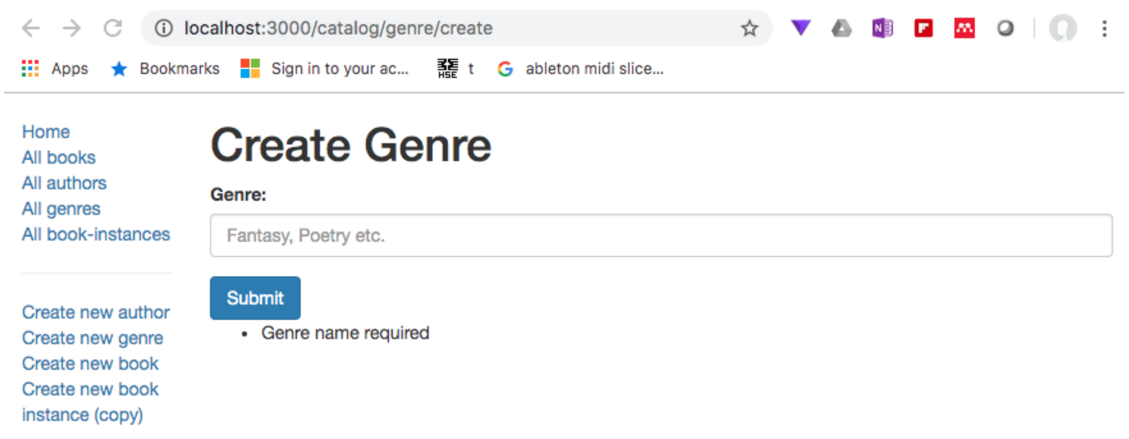
From the home page follow the Create new genre link or

localhost:3000/catalog/genre/create



Create Genre

Try to submit some a blank entry - genre name required prompt added.


localhost:3000/catalog/genre/create

Apps · Bookmarks · Sign in to your ac... · t · ableton midi slice...

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance (copy)

Create Genre

Genre:

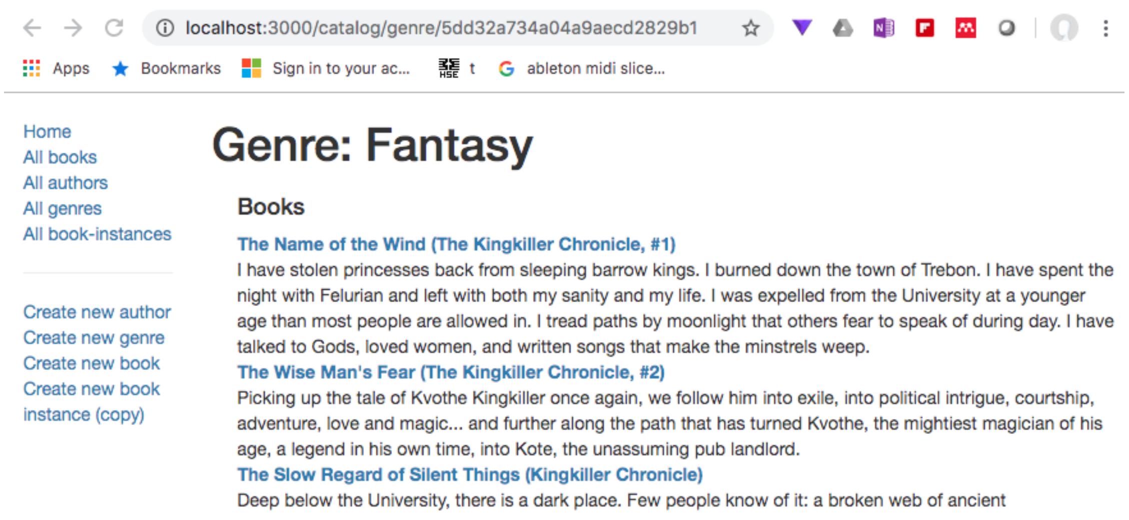
Fantasy, Poetry etc.

Submit

- Genre name required

Create Genre

Try to submit an existing genre, Fantasy, the list of fantasy books is displayed.


localhost:3000/catalog/genre/5dd32a734a04a9aecd2829b1

Apps · Bookmarks · Sign in to your ac... · t · ableton midi slice...

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance (copy)

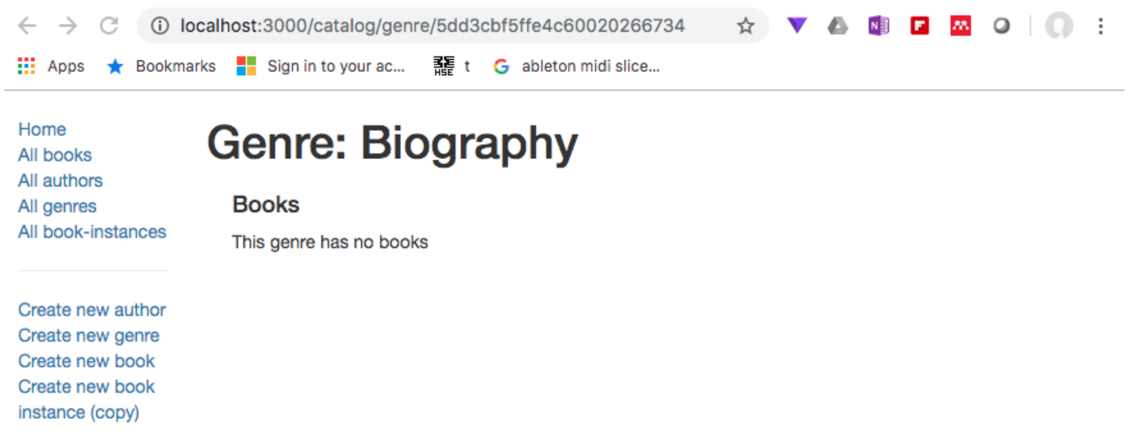
Genre: Fantasy

Books

The Name of the Wind (The Kingkiller Chronicle, #1)
I have stolen princesses back from sleeping barrow kings. I burned down the town of Trebon. I have spent the night with Felurian and left with both my sanity and my life. I was expelled from the University at a younger age than most people are allowed in. I tread paths by moonlight that others fear to speak of during day. I have talked to Gods, loved women, and written songs that make the minstrels weep.
The Wise Man's Fear (The Kingkiller Chronicle, #2)
Picking up the tale of Kvothe Kingkiller once again, we follow him into exile, into political intrigue, courtship, adventure, love and magic... and further along the path that has turned Kvothe, the mightiest magician of his age, a legend in his own time, into Kote, the unassuming pub landlord.
The Slow Regard of Silent Things (Kingkiller Chronicle)
Deep below the University, there is a dark place. Few people know of it: a broken web of ancient

Create Genre

Submit a new Genre Biography


localhost:3000/catalog/genre/5dd3cbf5ffe4c60020266734

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance (copy)

Genre: Biography

Books

This genre has no books

Create Genre

Within genre_form.pug, add required is true to the form element to add client side validation against and empty field. This is then validated belt and braces.

```
form(method='POST' action='')
    div.form-group
      label(for='name') Genre:
      input#name.form-control(type='text', placeholder='Fantasy, Poetry etc.'
name='name' value=(undefined===genre ? '' : genre.name), required='true')
    button.btn.btn-primary(type='submit') Submit
```

An attempt to add a blank genre is now caught in the browser.

Create Author

The pattern of working in creatin author is similar to genre create.

1. Use ‘required’ to import the functions needed to the controller file.
2. In the controller respond to a GET request by displaying the form using a pug template with a title.
3. In the controller respond to a POST request
    1. Validate fields
    2. Sanitize fields
    3. Process the request after validation and sanitization
        a. Extract validation errors from request

        b. If there are errors render form again with error messages

        c. If data is valid
            i. create a json object with trimmed data

            ii. save object to database
            iii. redirect to page to list of records

4. Create a pug template which matches the scheme of the database collection targetted.

Add the required validation and sanitization functions to controllers/authorController.js

```
var Author = require('../models/author');
var Book = require('../models/book');
var async = require('async');
const { body,validationResult } = require('express-validator/check');
const { sanitizeBody } = require('express-validator/filter');
```

In controllers/authorController.js replace code to handle author create GET request.

```
// Display Author create form on GET.
exports.author_create_get = function(req, res) {
    res.send('NOT IMPLEMENTED: Author create GET');
};
```

update with code to render the author create form using a pug template passing the title “Create Author”:

```
// Display Author create form on GET.
exports.author_create_get = function(req, res, next) {
    res.render('author_form', { title: 'Create Author'});
};
```

In controllers/authorController.js replace code to handle author create POST request.

```
// Handle Author create on POST.
exports.author_create_post = function(req, res) {
    res.send('NOT IMPLEMENTED: Author create POST');
};
```

with code to validate sanitize and process results.

```
// Handle Author create on POST.
exports.author_create_post = [

    // Validate fields.
    body('first_name').isLength({ min: 1 }).trim().withMessage('First name
must be specified.')
        .isAlphanumeric().withMessage('First name has non-alphanumeric
characters.'),
    body('family_name').isLength({ min: 1 }).trim().withMessage('Family name
must be specified.')
        .isAlphanumeric().withMessage('Family name has non-alphanumeric
characters.'),
    body('date_of_birth', 'Invalid date of birth').optional({ checkFalsy:
true }).isISO8601(),
    body('date_of_death', 'Invalid date of death').optional({ checkFalsy:
true }).isISO8601(),

    // Sanitize fields.
    sanitizeBody('first_name').escape(),
    sanitizeBody('family_name').escape(),
    sanitizeBody('date_of_birth').toDate(),
    sanitizeBody('date_of_death').toDate(),

    // Process request after validation and sanitization.
    (req, res, next) => {

        // Extract the validation errors from a request.
        const errors = validationResult(req);

        if (!errors.isEmpty()) {
            // There are errors. Render form again with sanitized
values/errors messages.
            res.render('author_form', { title: 'Create Author', author:
req.body, errors: errors.array() });
            return;
        }
        else {
            // Data from form is valid.

            // Create an Author object with escaped and trimmed data.
            var author = new Author(
                {
                    first_name: req.body.first_name,
                    family_name: req.body.family_name,
                    date_of_birth: req.body.date_of_birth,
                    date_of_death: req.body.date_of_death
                });
            author.save(function (err) {
                if (err) { return next(err); }
                // Successful - redirect to new author record.
                res.redirect(author.url);
            });
        }
    }
];
```

This code has not attempted to check if the author already exists so it is allowed to have multiple authors with the same name.

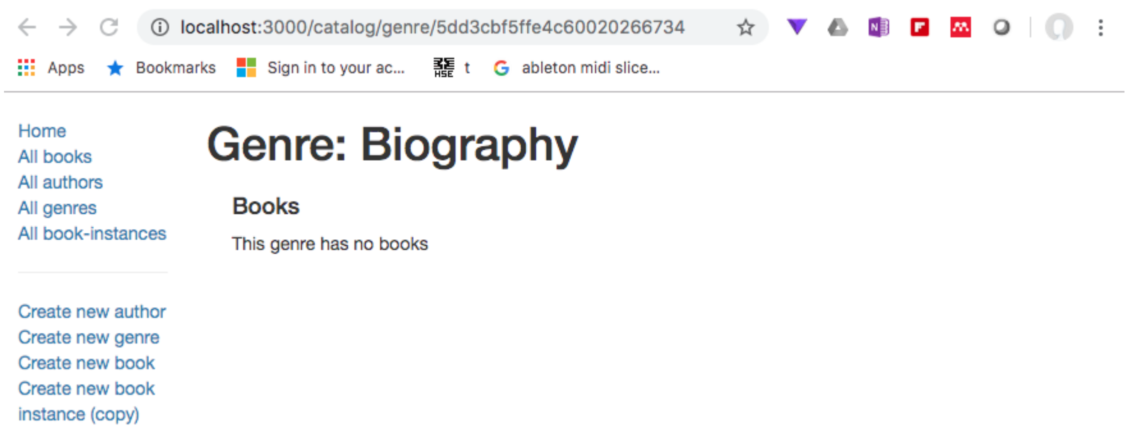
Create a pug template for the author create form in views/author_form.pug

```
extends layout

block content
  h1=title

  form(method='POST' action='')
    div.form-group
```

```
      label(for='first_name') First Name:
      input#first_name.form-control(type='text' placeholder='First name
(Christian) last' name='first_name' required='true' value=(undefined===author
? '' : author.first_name) )
      label(for='family_name') Family Name:
      input#family_name.form-control(type='text' placeholder='Family name
(surname)' name='family_name' required='true' value=(undefined===author ? ''
: author.family_name))
    div.form-group
      label(for='date_of_birth') Date of birth:
      input#date_of_birth.form-control(type='date' name='date_of_birth'
value=(undefined===author ? '' : author.date_of_birth) )
    div.form-group
      label(for='date_of_death') Date of death:
      input#date_of_death.form-control(type='date' name='date_of_death'
value=(undefined===author ? '' : author.date_of_death) )
    button.btn.btn-primary(type='submit') Submit
  if errors
    ul
      for error in errors
        li!= error.msg
```

Follow link to create new author or

http://localhost:3000/catalog/author/create



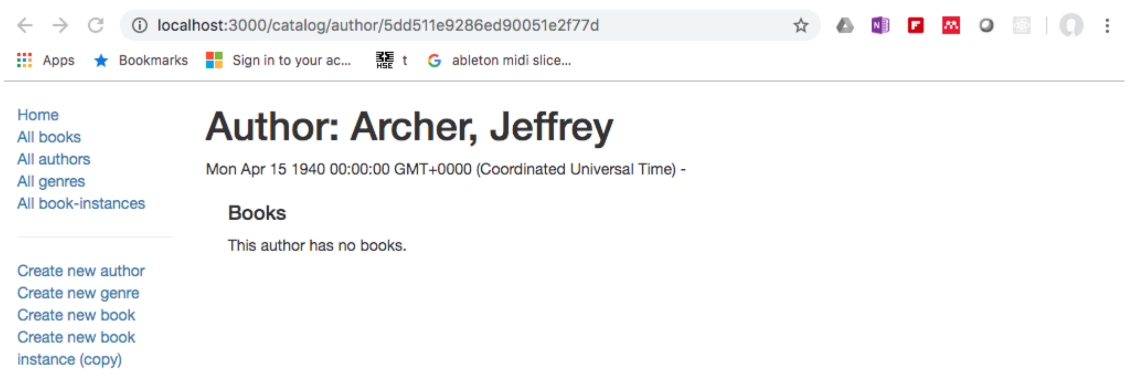
Genre: Biography

Books

This genre has no books

Create Genre

Submit details



Author: Archer, Jeffrey

Mon Apr 15 1940 00:00:00 GMT+0000 (Coordinated Universal Time) -

Books

This author has no books.

Create Genre

Create Book

The pattern of working in creating a book entry is similar to genre and author create, but will need to read author and genre details from the database to supply the appropriate id fields to the JSON object before the database entry is made.

1. Use 'required' to import the functions needed to the controller file.
2. In the controller respond to a GET request by displaying the form using a pug template with a title.
    1. Include details of all available authors and genres into the form using async.
3. In the controller respond to a POST request
    1. Validate fields
    2. Sanitize fields
    3. Process the request after validation and sanitization
        a. Extract validation errors from request

        b. If there are errors render form again with error messages

        c. If data is valid
            i. create a json object with trimmed data

            ii. save object to database
            iii. redirect to page to list of records
4. Create a pug template which matches the scheme of the database collection targetted.

In controllers/bookController.js add the functions needed by requiring them.

```
var Book = require('../models/book');
var Author = require('../models/author');
var Genre = require('../models/genre');
var BookInstance = require('../models/bookinstance');

var async = require('async');
```

```
const { body,validationResult } = require('express-validator/check');
const { sanitizeBody } = require('express-validator/filter');
```

In controllers/bookController.js replace the code for book_create_get.

```
// Display book create form on GET.
exports.book_create_get = function(req, res) {
    res.send('NOT IMPLEMENTED: Book create GET');
};
```

with code to display the book form through a put template.

```
// Display book create form on GET.
exports.book_create_get = function(req, res, next) {

    // Get all authors and genres, which we can use for adding to our book.
    async.parallel({
        authors: function(callback) {
            Author.find(callback);
        },
        genres: function(callback) {
            Genre.find(callback);
        },
    }, function(err, results) {
        if (err) { return next(err); }
        res.render('book_form', { title: 'Create Book', authors: 
results.authors, genres: results.genres });
    });

};
```

If there are no errors, the details which are passed on to the book_form include full author and genre content.

Replace the cotroller entry for a POST query to validate, sanitize and process a completed form. Replace:

```
// Handle book create on POST.
exports.book_create_post = function(req, res) {
    res.send('NOT IMPLEMENTED: Book create POST');
};
```

with code:

```
// Handle book create on POST.
exports.book_create_post = [
    // Convert the genre to an array.
    (req, res, next) => {
        if(!(req.body.genre instanceof Array)){
            if(typeof req.body.genre==='undefined')
            req.body.genre=[];
            else
            req.body.genre=new Array(req.body.genre);
        }
        next();
    },

    // Validate fields.
    body('title', 'Title must not be empty.').isLength({ min: 1 }).trim(),
    body('author', 'Author must not be empty.').isLength({ min: 1 }).trim(),
    body('summary', 'Summary must not be empty.').isLength({ min: 1 
}).trim(),
    body('isbn', 'ISBN must not be empty').isLength({ min: 1 }).trim(),

    // Sanitize fields (using wildcard).
    sanitizeBody('*').escape(),

    // Process request after validation and sanitization.
    (req, res, next) => {

        // Extract the validation errors from a request.
        const errors = validationResult(req);

        // Create a Book object with escaped and trimmed data.
        var book = new Book(
          { title: req.body.title,
            author: req.body.author,
            summary: req.body.summary,
            isbn: req.body.isbn,
            genre: req.body.genre
           });

        if (!errors.isEmpty()) {
            // There are errors. Render form again with sanitized 
values/error messages.

            // Get all authors and genres for form.
            async.parallel({
                authors: function(callback) {
                    Author.find(callback);
                },
                genres: function(callback) {
                    Genre.find(callback);
                },
            }, function(err, results) {
                if (err) { return next(err); }

                // Mark our selected genres as checked.
                for (let i = 0; i < results.genres.length; i++) {
                    if (book.genre.indexOf(results.genres[i]._id) > -1) {
                        results.genres[i].checked='true';
                    }
                }
                res.render('book_form', { title: 'Create
Book',authors:results.authors, genres:results.genres, book: book, errors:
errors.array() });
            });
            return;
        }
        else {
            // Data from form is valid. Save book.
            book.save(function (err) {
                if (err) { return next(err); }
                   //successful - redirect to new book record.
                   res.redirect(book.url);
                });
        }
    }
];
```

The sanitization is carried out for all fields using a wildcard rather than listing them individually.

A book only has one author but it may have one or more genres. Therefore the information for genre must be converted into a valid array, even if the entry for this field is empty.

The genres which are selected by the user are marked as checked.

Add the following code to the view file views/book_form.pug

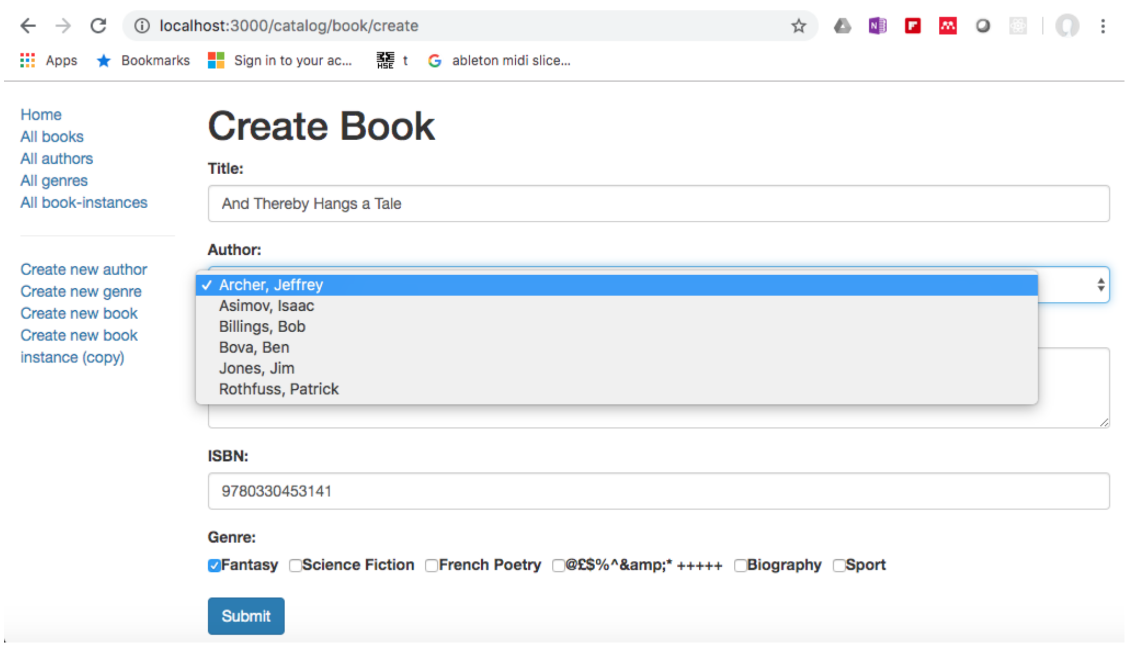
```
extends layout

block content
  h1= title

  form(method='POST' action='')
    div.form-group
      label(for='title') Title:
      input#title.form-control(type='text', placeholder='Name of book'
name='title' required='true' value=(undefined===book ? '' : book.title) )
    div.form-group
      label(for='author') Author:
      select#author.form-control(type='select', placeholder='Select author'
name='author' required='true' )
        - authors.sort(function(a, b) {let textA =
a.family_name.toUpperCase(); let textB = b.family_name.toUpperCase(); return
(textA < textB) ? -1 : (textA > textB) ? 1 : 0;});
        for author in authors
          if book
            option(value=author._id
selected=(author._id.toString()==book.author ? 'selected' : false) )
```

```
#{author.name}
          else
            option(value=author._id) #{author.name}
    div.form-group
      label(for='summary') Summary:
      textarea#summary.form-control(type='textarea', placeholder='Summary'
name='summary' rows='3' required='true') #{undefined===book ? '' :
book.summary}
    div.form-group
      label(for='isbn') ISBN:
      input#isbn.form-control(type='text', placeholder='ISBN13' name='isbn'
value=(undefined===book ? '' : book.isbn) required='true')
    div.form-group
      label Genre:
      div
        for genre in genres
          div(style='display: inline; padding-right:10px;')
            input.checkbox-input(type='checkbox', name='genre', id=genre._id,
value=genre._id, checked=genre.checked )
            label(for=genre._id) #{genre.name}
    button.btn.btn-primary(type='submit') Submit

  if errors
    ul
      for error in errors
        li!= error.msg
```

When this view is initiatied by the GET create book controler it displays the author list as a drop down select box and the genres as individual checkboxes.

The POST create book controller reads the contents and if there are no errors it redirects to the formed book url to show book details.

    http://localhost:3000/catalog/book/create



localhost:3000/catalog/book/create

Apps Bookmarks Sign in to your ac... t ableton midi slice...

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance (copy)

Create Book

Title: And Thereby Hangs a Tale

Author:
✓ Archer, Jeffrey
Asimov, Isaac
Billings, Bob
Bova, Ben
Jones, Jim
Rothfuss, Patrick

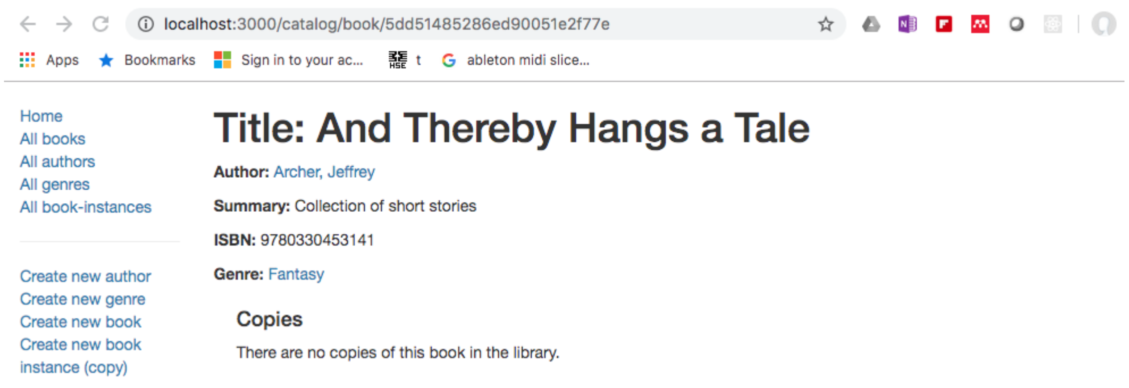
ISBN: 9780330453141

Genre: [✓ Fantasy] Science Fiction French Poetry @£$%^&amp;* +++++ Biography Sport

Submit

Create Genre

Submit details



localhost:3000/catalog/book/5dd51485286ed90051e2f77e

Apps Bookmarks Sign in to your ac... t ableton midi slice...

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance (copy)

Title: And Thereby Hangs a Tale

Author: Archer, Jeffrey

Summary: Collection of short stories

ISBN: 9780330453141

Genre: Fantasy

Copies

There are no copies of this book in the library.

Create Genre

Although a new book has been created there are no instances of it in the local_library.

Create BookInstance

The pattern of working in creating a bookInstance entry is similar to genre and author create, but will need to book details from the database to supply the appropriate id fields to the JSON object before the database entry is made.

1. Use ‘required’ to import the functions needed to the controller file.
2. In the controller respond to a GET request by displaying the form using a pug template with a title.
    1. Include details of all available book titles into the form using find.

3. In the controller respond to a POST request
    1. Validate fields
    2. Sanitize fields
    3. Process the request after validation and sanitization
        a. Extract validation errors from request

        b. If there are errors render form again with error messages

        c. If data is valid
            i. create a json object with trimmed data

            ii. save object to database
            iii. redirect to page to list of records
4. Create a pug template which matches the scheme of the database collection targetted.

In controllers/bookinstanceController.js add the functions needed by requiring them. Add a requirement for the book model.

```
var BookInstance = require('../models/bookinstance');
var Book = require('../models/book');
const { body,validationResult } = require('express-validator/check');
const { sanitizeBody } = require('express-validator/filter');
```

Replace the book Instance create get code:

```
// Display BookInstance create form on GET.
exports.bookinstance_create_get = function(req, res) {
    res.send('NOT IMPLEMENTED: BookInstance create GET');
};
```

with code to find the available book titles and pass these to the bookinstance_form view with the form title:

```
// Display BookInstance create form on GET.
exports.bookinstance_create_get = function(req, res, next) {

    Book.find({},'title')
    .exec(function (err, books) {
      if (err) { return next(err); }
      // Successful, so render.
      res.render('bookinstance_form', {title: 'Create BookInstance',
book_list: books});
    });

};
```

Replace the code for bookinstance create on POST

```
// Handle BookInstance create on POST.
exports.bookinstance_create_post = function(req, res) {
    res.send('NOT IMPLEMENTED: BookInstance create POST');
};
```

with code to validate, sanitize and process the request.

```
// Handle BookInstance create on POST.
exports.bookinstance_create_post = [

    // Validate fields.
    body('book', 'Book must be specified').isLength({ min: 1 }).trim(),
    body('imprint', 'Imprint must be specified').isLength({ min: 1 }).trim(),
    body('due_back', 'Invalid date').optional({ checkFalsy: true
}).isISO8601(),

    // Sanitize fields.
    sanitizeBody('book').escape(),
    sanitizeBody('imprint').escape(),
    sanitizeBody('status').trim().escape(),
    sanitizeBody('due_back').toDate(),

    // Process request after validation and sanitization.
    (req, res, next) => {

        // Extract the validation errors from a request.
        const errors = validationResult(req);

        // Create a BookInstance object with escaped and trimmed data.
        var bookinstance = new BookInstance(
          { book: req.body.book,
            imprint: req.body.imprint,
            status: req.body.status,
            due_back: req.body.due_back
           });

        if (!errors.isEmpty()) {
            // There are errors. Render form again with sanitized values and
error messages.
            Book.find({},'title')
                .exec(function (err, books) {
                    if (err) { return next(err); }
                    // Successful, so render.
                    res.render('bookinstance_form', { title: 'Create
BookInstance', book_list: books, selected_book: bookinstance.book._id ,
errors: errors.array(), bookinstance: bookinstance });
            });
            return;
        }
        else {
            // Data from form is valid.
            bookinstance.save(function (err) {
                if (err) { return next(err); }
                   // Successful - redirect to new record.
                   res.redirect(bookinstance.url);
                });
        }
    }
];
```

If the form contains a valid entry the bookinstance is passed to the database and the user is returned to the bookinstance url.

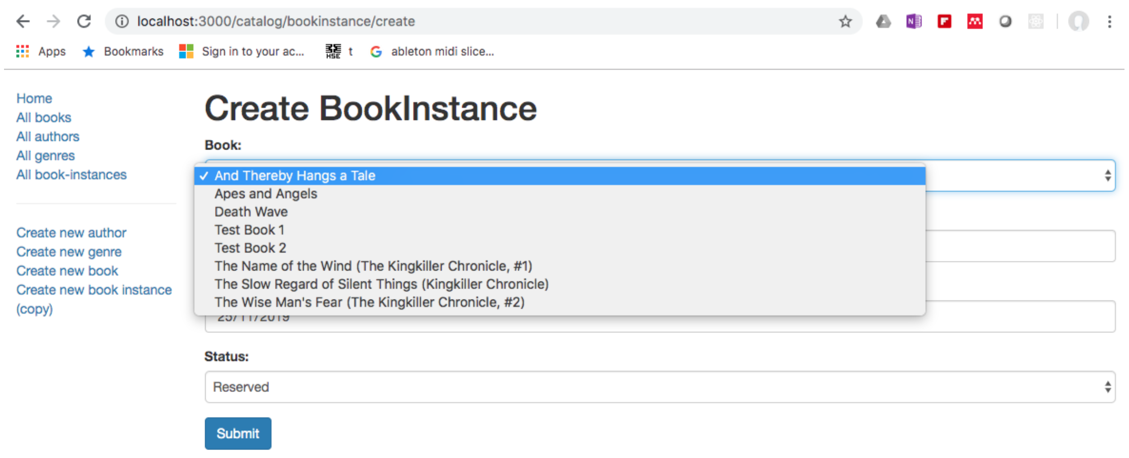
Create the form view in views/bookinstance_form.pug

```
extends layout

block content
  h1=title

  form(method='POST' action='')
    div.form-group
      label(for='book') Book:
      select#book.form-control(type='select' placeholder='Select book'
name='book' required='true')
        - book_list.sort(function(a, b) {let textA = a.title.toUpperCase();
let textB = b.title.toUpperCase(); return (textA < textB) ? -1 : (textA >
textB) ? 1 : 0;});
        for book in book_list
          if bookinstance
            option(value=book._id
selected=(bookinstance.book.toString()==book._id.toString() ? 'selected' :
false)) #{book.title}
          else
            option(value=book._id) #{book.title}

    div.form-group
      label(for='imprint') Imprint:
      input#imprint.form-control(type='text' placeholder='Publisher and date
information' name='imprint' required='true' value=(undefined===bookinstance ?
'' : bookinstance.imprint))
    div.form-group
      label(for='due_back') Date when book available:
      input#due_back.form-control(type='date' name='due_back'
value=(undefined===bookinstance ? '' : bookinstance.due_back))

    div.form-group
      label(for='status') Status:
      select#status.form-control(type='select' placeholder='Select status'
name='status' required='true')
```

```
        option(value='Maintenance') Maintenance
        option(value='Available') Available
        option(value='Loaned') Loaned
        option(value='Reserved') Reserved

    button.btn.btn-primary(type='submit') Submit

  if errors
    ul
      for error in errors
        li!= error.msg
```

Note that the Status Select drop down menu options are hard coded. This is ok if there are to be a small number of fixed status available, however the example could be developed by adding a collection of bookStates.

http://localhost:3000/catalog/bookinstance/create



```
localhost:3000/catalog/bookinstance/create
```

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance
(copy)

Create BookInstance

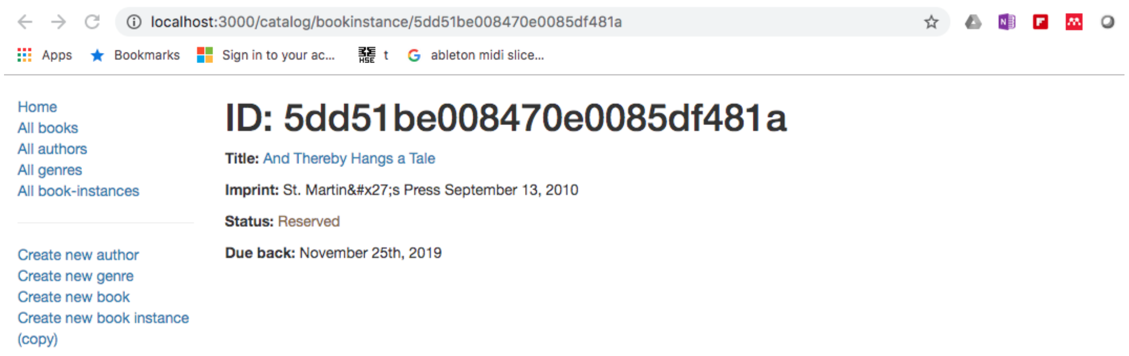
Book:
✓ And Thereby Hangs a Tale
Apes and Angels
Death Wave
Test Book 1
Test Book 2
The Name of the Wind (The Kingkiller Chronicle, #1)
The Slow Regard of Silent Things (Kingkiller Chronicle)
The Wise Man's Fear (The Kingkiller Chronicle, #2)

Status: Reserved

Submit

Create Genre

Submit details



```
localhost:3000/catalog/bookinstance/5dd51be008470e0085df481a
```

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance
(copy)

ID: 5dd51be008470e0085df481a

Title: And Thereby Hangs a Tale

Imprint: St. Martin&#x27;s Press September 13, 2010

Status: Reserved

Due back: November 25th, 2019

Create Genre

A small detail to note is that the apostrophe is being listed as &#x27; which cold corrected.

That concludes the form entries for creating new content. Next step is to build forms for Deletion.

Delete Author

1. At the foot of each author detail page add a link to delete the author which will call author.url+'/delete' with a GET request.
2. In the controller respond to a GET request by displaying the form using a pug template with a title.
    1. If the author has assosciated books display these in a pug view. Books must be deleted before author is removed.
    2. If there are no books associated with the author display details and request confirmation of deletion as a POST request.
3. In the controller respond to a POST request
    1. Request author details and book details async
    2. Process the request
        a. Recheck and if author has associated books do not allow deletion but return to the author delete form.
        b. If author has no associated books, delete and redirect to catalog/authors list.
4. Create a pug template for the authors delete form.

Add a link to delete code to the foot of author_detail.pug

```
extends layout

block content

  h1 Author: #{author.name}
  p #{author.date_of_birth} - #{author.date_of_death}

  div(style='margin-left:20px;margin-top:20px')

    h4 Books

    dl
      each book in author_books
        dt
          a(href=book.url) #{book.title}
        dd #{book.summary}

      else
        p This author has no books.
hr
p
  a(href=author.url+'/delete') Delete author
```

Replace Display Author delete form on GET:

```
// Display Author delete form on GET.
exports.author_delete_get = function(req, res) {
    res.send('NOT IMPLEMENTED: Author delete GET');
};
```

with code:

```
// Display Author delete form on GET.
exports.author_delete_get = function(req, res, next) {

    async.parallel({
        author: function(callback) {
            Author.findById(req.params.id).exec(callback)
        },
        authors_books: function(callback) {
          Book.find({ 'author': req.params.id }).exec(callback)
        },
    }, function(err, results) {
        if (err) { return next(err); }
        if (results.author==null) { // No results.
            res.redirect('/catalog/authors');
        }
        // Successful, so render.
        res.render('author_delete', { title: 'Delete Author', author:
results.author, author_books: results.authors_books } );
    });

};
```

TIf an author is found the name is shown in views/author_delete.pug.

```
extends layout

block content
  h1 #{title}: #{author.name}
  p= author.lifespan

  if author_books.length

    p #[strong Delete the following books before attempting to delete this
author.]

    div(style='margin-left:20px;margin-top:20px')

      h4 Books

      dl
      each book in author_books
        dt
          a(href=book.url) #{book.title}
        dd #{book.summary}

  else
    p Do you really want to delete this Author?

    form(method='POST' action='')
      div.form-group
        input#authorid.form-control(type='hidden',name='authorid',
required='true', value=author._id )

      button.btn.btn-primary(type='submit') Delete
```

This view presents an error over the author.lifespan, this is because the virtual lifespan in the author model depends on the difference between recorded birth and death dates, however some of these fields are blank. The controller will need to be ammended using if then else to return a blank space if one of these fields is blank.

A short form of if then else is:

```
condition ? statementIfTrue : statementIfFalse ;
```

At the same time moment can be used to neaten up the dates so that the lifetime can be expressed in years.

Within models/author.js require the moment module.

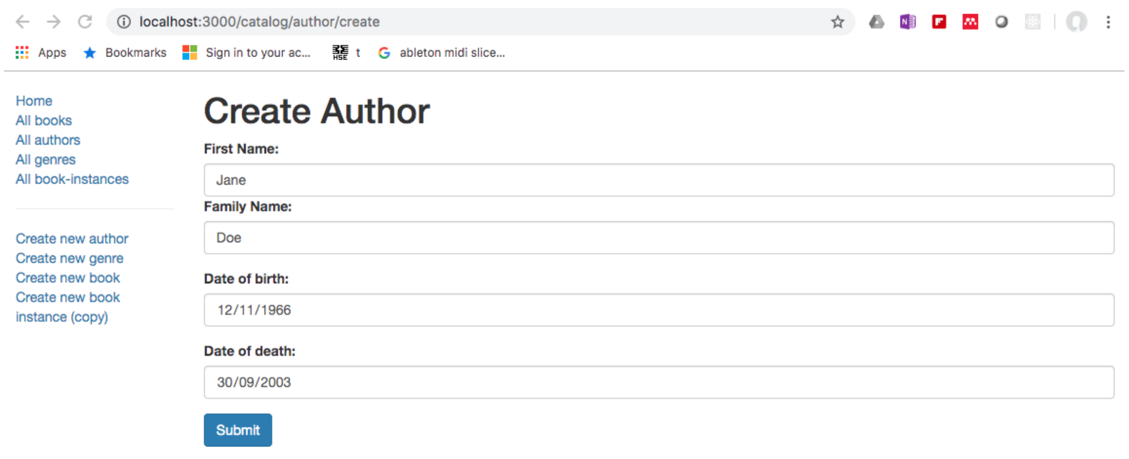
```
  var mongoose = require('mongoose');
  var moment = require('moment');
```

within the virtual authors lifespan, comment out the original lifespan calculation and replace this with a moment formatted calculation which is returned if both date_of_death and date_of_life fields are not blank. If either field is blank a blank character is returned.

```
  // Virtual for author's lifespan
AuthorSchema
.virtual('lifespan')
.get(function () {
  //return (this.date_of_death.getYear() -
this.date_of_birth.getYear()).toString();
  return (this.date_of_death && this.date_of_death)
    ? moment((this.date_of_death.getYear() - this.date_of_birth.getYear())):
'';

});
```

To test this, create a test author entry and then delete it.

        http://localhost:3000/catalog/author/create


localhost:3000/catalog/author/create

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance (copy)

Create Author

First Name: Jane

Family Name: Doe

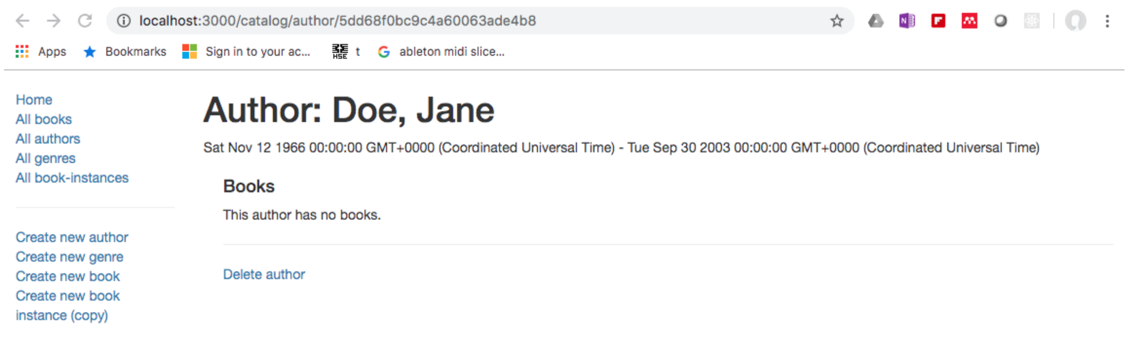
Date of birth: 12/11/1966

Date of death: 30/09/2003

Submit

Create Genre

Submit


localhost:3000/catalog/author/5dd68f0bc9c4a60063ade4b8

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance (copy)

Author: Doe, Jane

Sat Nov 12 1966 00:00:00 GMT+0000 (Coordinated Universal Time) - Tue Sep 30 2003 00:00:00 GMT+0000 (Coordinated Universal Time)

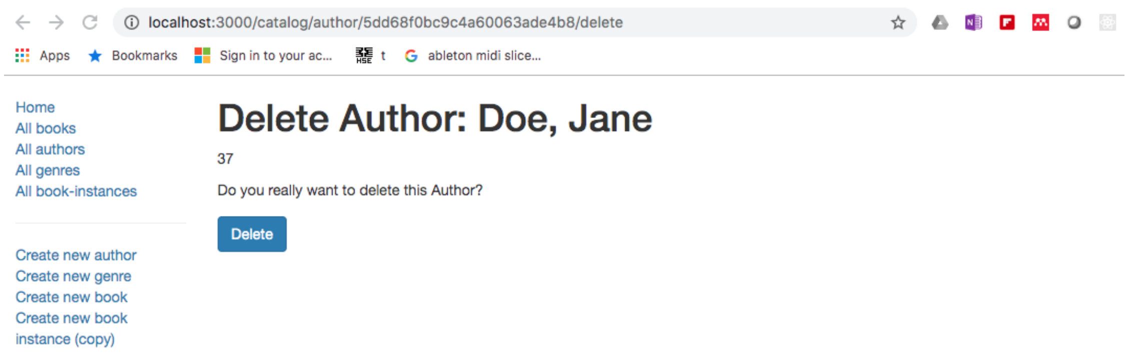
Books

This author has no books.

Delete author

Create Genre

Author Delete


localhost:3000/catalog/author/5dd68f0bc9c4a60063ade4b8/delete

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance (copy)

Delete Author: Doe, Jane

37

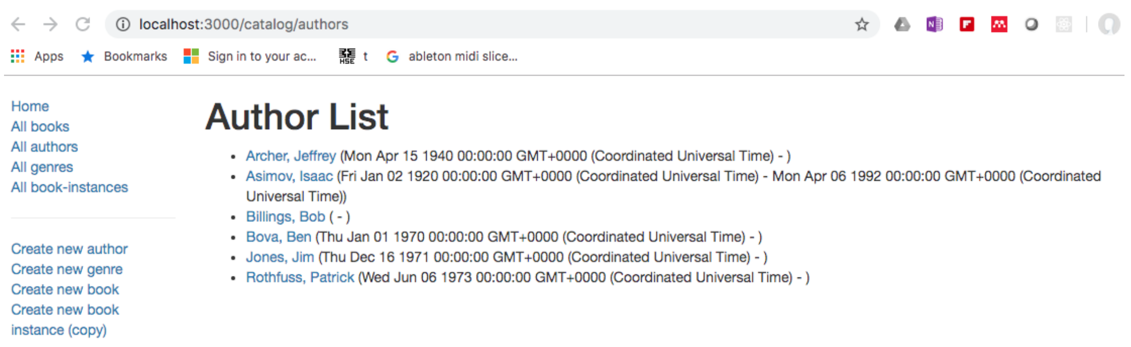
Do you really want to delete this Author?

Delete

Create Genre

Note life span calculation.

Delete.



Author List

- Archer, Jeffrey (Mon Apr 15 1940 00:00:00 GMT+0000 (Coordinated Universal Time) -)
- Asimov, Isaac (Fri Jan 02 1920 00:00:00 GMT+0000 (Coordinated Universal Time) - Mon Apr 06 1992 00:00:00 GMT+0000 (Coordinated Universal Time))
- Billings, Bob (-)
- Bova, Ben (Thu Jan 01 1970 00:00:00 GMT+0000 (Coordinated Universal Time) -)
- Jones, Jim (Thu Dec 16 1971 00:00:00 GMT+0000 (Coordinated Universal Time) -)
- Rothfuss, Patrick (Wed Jun 06 1973 00:00:00 GMT+0000 (Coordinated Universal Time) -)

Create Genre

##Futher work

Develop Delete forms for for the Book, BookInstance, and Genre models.

Update Book Form

The create, retrieve and delete processes have been implimented which just leaves update.

Updating is like creating except that the form must be filled in from the current database entry.

1. In the controller respond to a GET request by displaying the form using a pug template with a title.
    1. Include details of all available book titles into the form using find.
2. In the controller respond to a POST request
    1. Validate fields
    2. Sanitize fields
    3. Process the request after validation and sanitization
        a. Extract validation errors from request

        b. If there are errors render form again with error messages

        c. If data is valid
            i. create a json object with trimmed data

            ii. save object to database
            iii. redirect to page to list of records
3. create views for forms

In controllers/bookController.js, replace:

```
// Display book update form on GET.
exports.book_update_get = function(req, res) {
    res.send('NOT IMPLEMENTED: Book update GET');
};
```

with code:

```
// Display book update form on GET.
exports.book_update_get = function(req, res, next) {

    // Get book, authors and genres for form.
    async.parallel({
        book: function(callback) {

Book.findById(req.params.id).populate('author').populate('genre').exec(callba
ck);
        },
        authors: function(callback) {
            Author.find(callback);
        },
        genres: function(callback) {
            Genre.find(callback);
        },
        }, function(err, results) {
            if (err) { return next(err); }
            if (results.book==null) { // No results.
                var err = new Error('Book not found');
                err.status = 404;
                return next(err);
            }
            // Success.
            // Mark our selected genres as checked.
            for (var all_g_iter = 0; all_g_iter < results.genres.length;
all_g_iter++) {
                for (var book_g_iter = 0; book_g_iter <
results.book.genre.length; book_g_iter++) {
                    if
(results.genres[all_g_iter]._id.toString()==results.book.genre[book_g_iter]._
id.toString()) {
                        results.genres[all_g_iter].checked='true';
                    }
                }
            }
            res.render('book_form', { title: 'Update Book', authors:
results.authors, genres: results.genres, book: results.book });
        });

};
```

This is called by a GET request passing the id of a particular book as a parameter. The result, if succesful is data which is passed to the Update Book form.

The completed form will submit to update book on POST. Replace :

```
// Handle book update on POST.
exports.book_update_post = function(req, res) {
    res.send('NOT IMPLEMENTED: Book update POST');
};
```

with code:

```
// Handle book update on POST.
exports.book_update_post = [

    // Convert the genre to an array
    (req, res, next) => {
        if(!(req.body.genre instanceof Array)){
            if(typeof req.body.genre==='undefined')
            req.body.genre=[];
            else
            req.body.genre=new Array(req.body.genre);
        }
        next();
    },

    // Validate fields.
    body('title', 'Title must not be empty.').isLength({ min: 1 }).trim(),
    body('author', 'Author must not be empty.').isLength({ min: 1 }).trim(),
    body('summary', 'Summary must not be empty.').isLength({ min: 1
}).trim(),
    body('isbn', 'ISBN must not be empty').isLength({ min: 1 }).trim(),

    // Sanitize fields.
    sanitizeBody('title').escape(),
    sanitizeBody('author').escape(),
    sanitizeBody('summary').escape(),
    sanitizeBody('isbn').escape(),
    sanitizeBody('genre.*').escape(),

    // Process request after validation and sanitization.
    (req, res, next) => {

        // Extract the validation errors from a request.
        const errors = validationResult(req);

        // Create a Book object with escaped/trimmed data and old id.
        var book = new Book(
          { title: req.body.title,
            author: req.body.author,
            summary: req.body.summary,
            isbn: req.body.isbn,
            genre: (typeof req.body.genre==='undefined') ? [] :
```

```
req.body.genre,
            _id:req.params.id //This is required, or a new ID will be
assigned!
           });

        if (!errors.isEmpty()) {
            // There are errors. Render form again with sanitized
values/error messages.

            // Get all authors and genres for form.
            async.parallel({
                authors: function(callback) {
                    Author.find(callback);
                },
                genres: function(callback) {
                    Genre.find(callback);
                },
            }, function(err, results) {
                if (err) { return next(err); }

                // Mark our selected genres as checked.
                for (let i = 0; i < results.genres.length; i++) {
                    if (book.genre.indexOf(results.genres[i]._id) > -1) {
                        results.genres[i].checked='true';
                    }
                }
                res.render('book_form', { title: 'Update Book',authors:
results.authors, genres: results.genres, book: book, errors: errors.array()
});
            });
            return;
        }
        else {
            // Data from form is valid. Update the record.
            Book.findByIdAndUpdate(req.params.id, book, {}, function
(err,thebook) {
                if (err) { return next(err); }
                   // Successful - redirect to book detail page.
                   res.redirect(thebook.url);
                });
        }
    }
];
```

Valid data is passed into the database and the book detail is displayed.

Modify the file views/book_form.pug so that it can deal with both create and update replacing:

```
  div.form-group
   label(for='author') Author:
   select#author.form-control(type='select' placeholder='Select author'
name='author' required='true' )
    for author in authors
     if book
      //- Handle GET form, where book.author is an object, and POST
form, where it is a string.
      option(
       value=author._id
       selected=(
        author._id.toString()==book.author._id
        || author._id.toString()==book.author
       ) ? 'selected' : false
      ) #{author.name}
     else
      option(value=author._id) #{author.name}
```

with the code:

```
 div.form-group
   label(for='author') Author:
   select#author.form-control(type='select' placeholder='Select author'
name='author' required='true' )
    for author in authors
     if book
      //- Handle GET form, where book.author is an object, and POST
form, where it is a string.
      option(
       value=author._id
       selected=(
        author._id.toString()==book.author._id
        || author._id.toString()==book.author
       ) ? 'selected' : false
      ) #{author.name}
     else
      option(value=author._id) #{author.name}
```

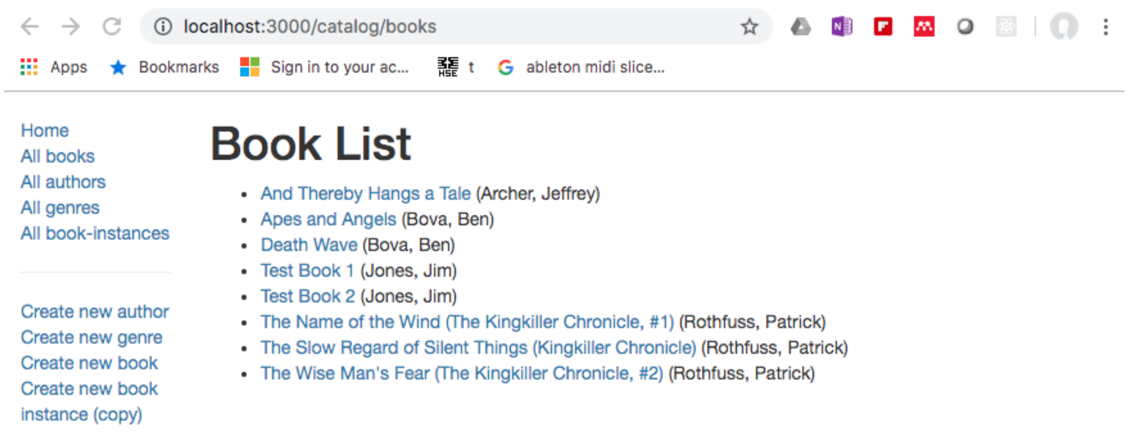
Add adelete and update buttons to the bottom of views/book_detail.pug

```
hr
 p
  a(href=book.url+'/delete') Delete Book
 p
  a(href=book.url+'/update') Update Book
```

Before updating a book with a new author, the author must be created, and if needed, a new genre.

Now try to update a book.

http://localhost:3000/catalog/books

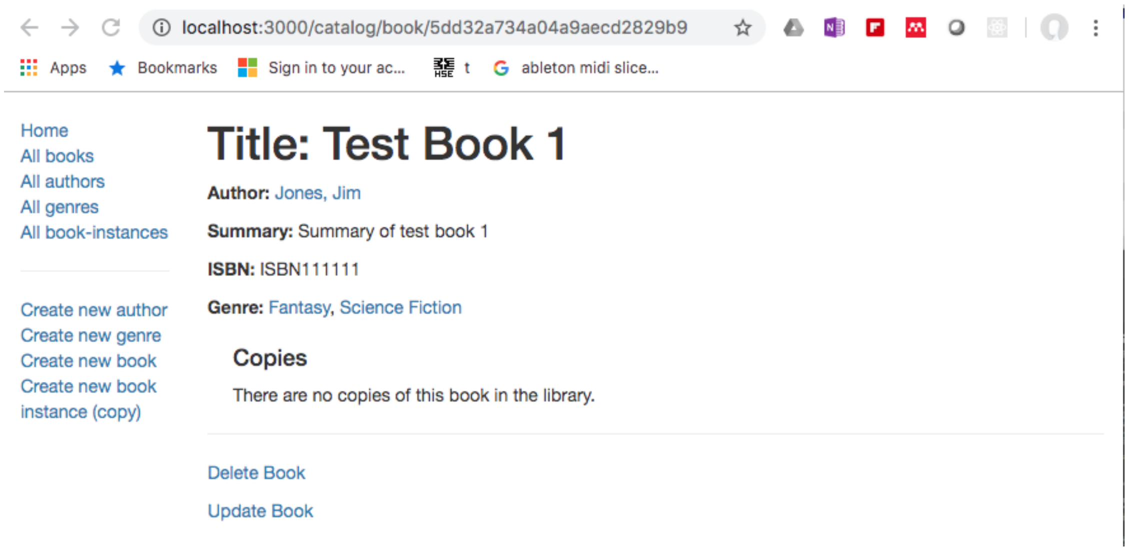


Book List

- And Thereby Hangs a Tale (Archer, Jeffrey)
- Apes and Angels (Bova, Ben)
- Death Wave (Bova, Ben)
- Test Book 1 (Jones, Jim)
- Test Book 2 (Jones, Jim)
- The Name of the Wind (The Kingkiller Chronicle, #1) (Rothfuss, Patrick)
- The Slow Regard of Silent Things (Kingkiller Chronicle) (Rothfuss, Patrick)
- The Wise Man's Fear (The Kingkiller Chronicle, #2) (Rothfuss, Patrick)

Create Genre

Navigate to Test book one



Title: Test Book 1

Author: Jones, Jim

Summary: Summary of test book 1

ISBN: ISBN111111

Genre: Fantasy, Science Fiction

Copies

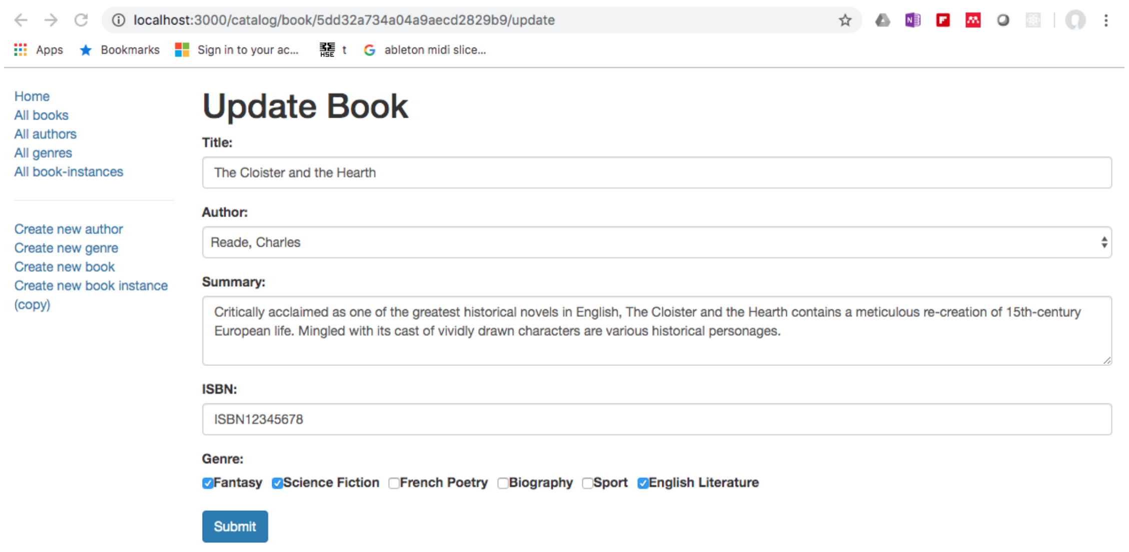
There are no copies of this book in the library.

Delete Book

Update Book

Create Genre

Follow the update Book link.



localhost:3000/catalog/book/5dd32a734a04a9aecd2829b9/update

Apps Bookmarks Sign in to your ac... t ableton midi slice...

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance
(copy)

Update Book

Title:

The Cloister and the Hearth

Author:

Reade, Charles

Summary:

Critically acclaimed as one of the greatest historical novels in English, The Cloister and the Hearth contains a meticulous re-creation of 15th-century European life. Mingled with its cast of vividly drawn characters are various historical personages.

ISBN:

ISBN12345678

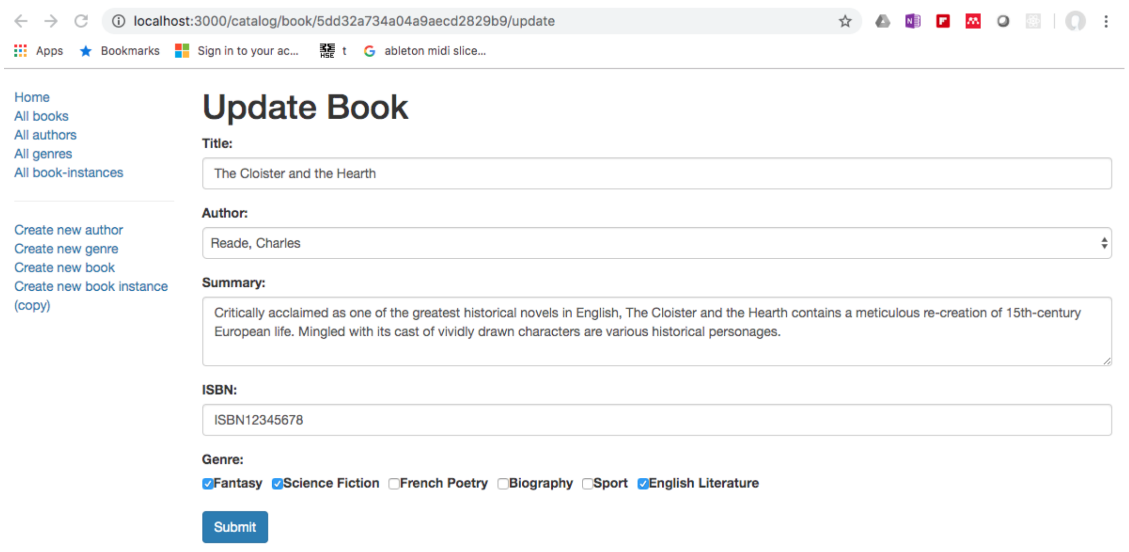
Genre:

[✓ Fantasy] [✓ Science Fiction] French Poetry Biography Sport [✓ English Literature]

Submit

Create Genre

The updated detail is displayed.



localhost:3000/catalog/book/5dd32a734a04a9aecd2829b9/update

Apps Bookmarks Sign in to your ac... t ableton midi slice...

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance
(copy)

Update Book

Title:

The Cloister and the Hearth

Author:

Reade, Charles

Summary:

Critically acclaimed as one of the greatest historical novels in English, The Cloister and the Hearth contains a meticulous re-creation of 15th-century European life. Mingled with its cast of vividly drawn characters are various historical personages.

ISBN:

ISBN12345678

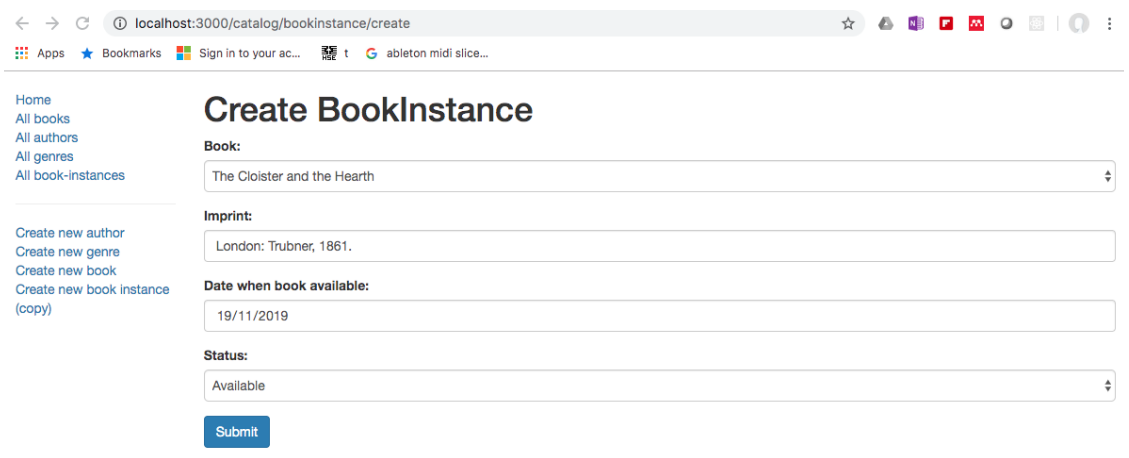
Genre:

[✓ Fantasy] [✓ Science Fiction] French Poetry Biography Sport [✓ English Literature]

Submit

Create Genre

We may as well create an instance of the book!


localhost:3000/catalog/bookinstance/create

Apps Bookmarks Sign in to your ac... t ableton midi slice...

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance (copy)

Create BookInstance

Book:
The Cloister and the Hearth

Imprint:
London: Trubner, 1861.

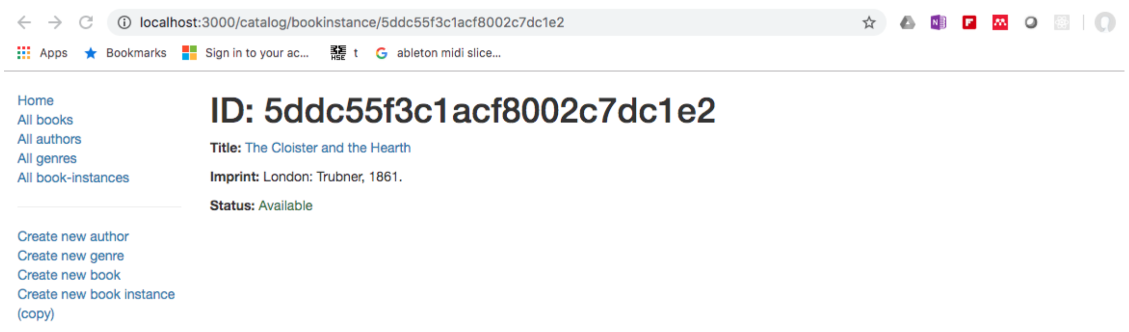
Date when book available:
19/11/2019

Status:
Available

Submit

Create Genre

Submit.


localhost:3000/catalog/bookinstance/5ddc55f3c1acf8002c7dc1e2

Apps Bookmarks Sign in to your ac... t ableton midi slice...

Home
All books
All authors
All genres
All book-instances

Create new author
Create new genre
Create new book
Create new book instance (copy)

ID: 5ddc55f3c1acf8002c7dc1e2

Title: The Cloister and the Hearth

Imprint: London: Trubner, 1861.

Status: Available

Create Genre

Exercise

It is left as an exercise to complete other delete and update pages.