Comic Collector Documentation

Release 1.0a1

Ben Cole

October 13, 2015

Contents

Contents:

CHAPTER 1

Features

1.1 Catalogue

The catalogue is the collection of books the Comic Collector knows about. They are stored in a database.

Different types of books can be represented by extending the base *Book* model

Todo

More information about how the catalogue is represented in the UI

1.2 Global Search

Currently there is only a 3rd party search view. The intention is to make on universal search that covers books in the current user's *lists*, books in the *Catalogue*, and books from 3rd party search, in that order

CHAPTER 2

User Stories

This page is aimed at being a reference for developers of the features intended to be implemented from a user's perspective

There are 3 types of users of the site:

1. *Curator* - A curator of the *Catalogue* of books.
2. *Collector* - Someone who is collecting books.
3. *Gifter* - Someone who wishes to buy a *Collector* a gift.

2.1 Authentication

django-allauth will be used to used to ease the signup process, and give gifters as easy as possible an experience

2.2 Curator

As an Curator ...

- I can search a 3rd party for books
  - I can add relevant search results to the *Catalogue*
  - I can use search results to update existing books in the *Catalogue*
- I can search the *Catalogue*
  - I can use a result as the basis of a search against a 3rd party in order to update missing information
- I can access the Django admin for any further maintenance

2.3 Collector

As a collector ...

- I can search the *Catalogue* for books [1]
  - I can add results to one or more of my *List Models*

[1] Initially 2 separate search pages will exist; one for the *Catalogue* and one for 3rd party search. The later intention is to create a *Global Search*.

- I can search a 3rd party for books
  - I can add results to the *Catalogue* (no editing allowed)
  - I can add results directly to my *List Models*
- I can get a list of my *List Models*
  - I can drill-down into a list to see its contents
- I can manage one of my *List Models*
  - I can add/remove books to the list
  - I can change the status of a book in the list
  - I can re-order the books in the list

2.4 Gifter

As a gifter ...

- I can search for a *Wishlist* by owner's name or email
- I can follow a link of a *Wishlist* to a 3rd party online store
- I can mark an item on a *Wishlist* as purchased elsewhere

CHAPTER 3

Data Structures

This project (currently) [1] consists of the following models:

3.1 Book Models

3.1.1 Book

Generic book model with ISBN, title etc

3.1.2 Volume

A “graphic novel” collecting multiple *Issues* from a *Series*

Extends *Book* adding multiple extra fields, most importantly a list the collected *Issues*

3.1.3 Event Book

A large book that collects most if not all of the *Issues* from an *Event*

Extends *Book* adding 2 fields *Event* and *Issues* [2]

3.1.4 Issues

Although technically a book in real life, for simplicity this is a basic model that just tracks an issue number with a *Series*

3.1.5 Series

A very simple model with just the name of the series

3.1.6 Event

Another simple model with just the name of the event

[1] The intention is to later make this more of a generic collection application

[2] This should probably extend *Volume* instead

3.2 List Models

3.2.1 Collection

All the books in a user's collection

3.2.2 Reading Order

The order in which the user intends to read the book Extends *Collection* adding order and read status fields

3.2.3 Wishlist

The books the user wishes to buy (or have bought for them) Extends *Collection* adding order and purchased fields

Note: The purchase state would be hidden from the user, but would prevent a *Gifter* from buying a book already purchased. If the user themselves were to purchase the book, another flow would take place, removing the book from the wishlist and adding it to the collection

CHAPTER 4

Indices and tables

- genindex
- modindex
- search