Building a data-model

Pattern #1

Almost any application in OutSystems will require developers to create entities to represent business concepts that are stored in a relational database. If these concepts are not meant to be changed at runtime (like lookup tables or enumerations), developers can create static entities and define existing records at design time. Business concepts tend to be related, and developers can do it by establishing relationships between two or more concepts.

The first part of this exercise requires you to create an entity and a static entity, along with the screens that allow users to manipulate them. In the second part of the exercise it will require you to establish a relationship between them and update the screens so that association can be displayed.

KEYWORDS: **data-model**, **entity**, **static entity**, **scaffolding**, **relationship**, **dropdown**, **filter**

Develop the MyLibrary application

The MyLibrary application allows users to build a catalog of books. The two main concepts are a **book** and its **genre**. When going through the catalog, the application will allow the user to search by book title, and will also allow the user to filter books by a specific genre.

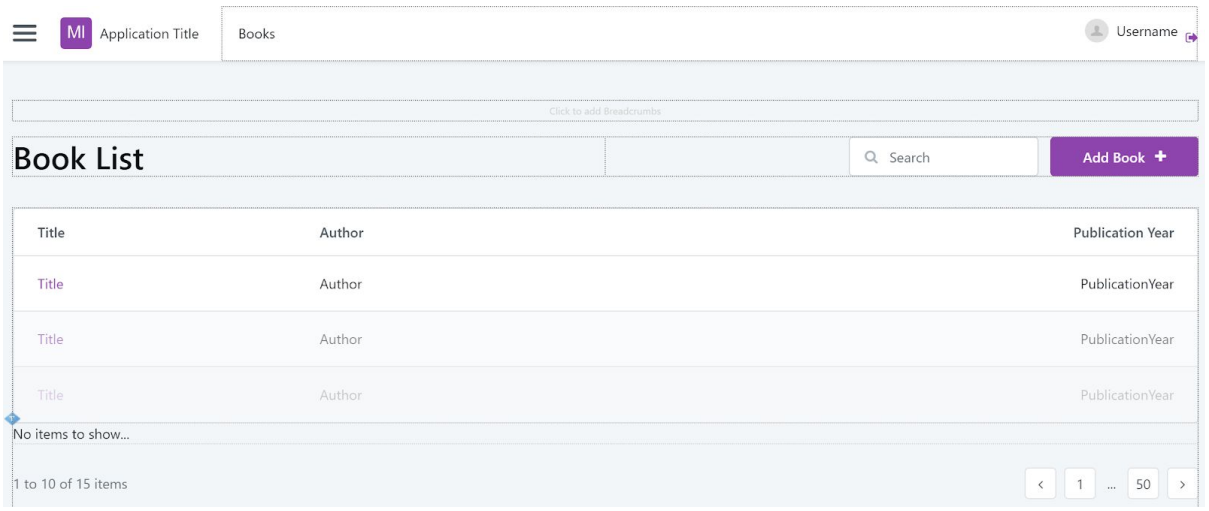
Part I - Creating the base data-model and application screens

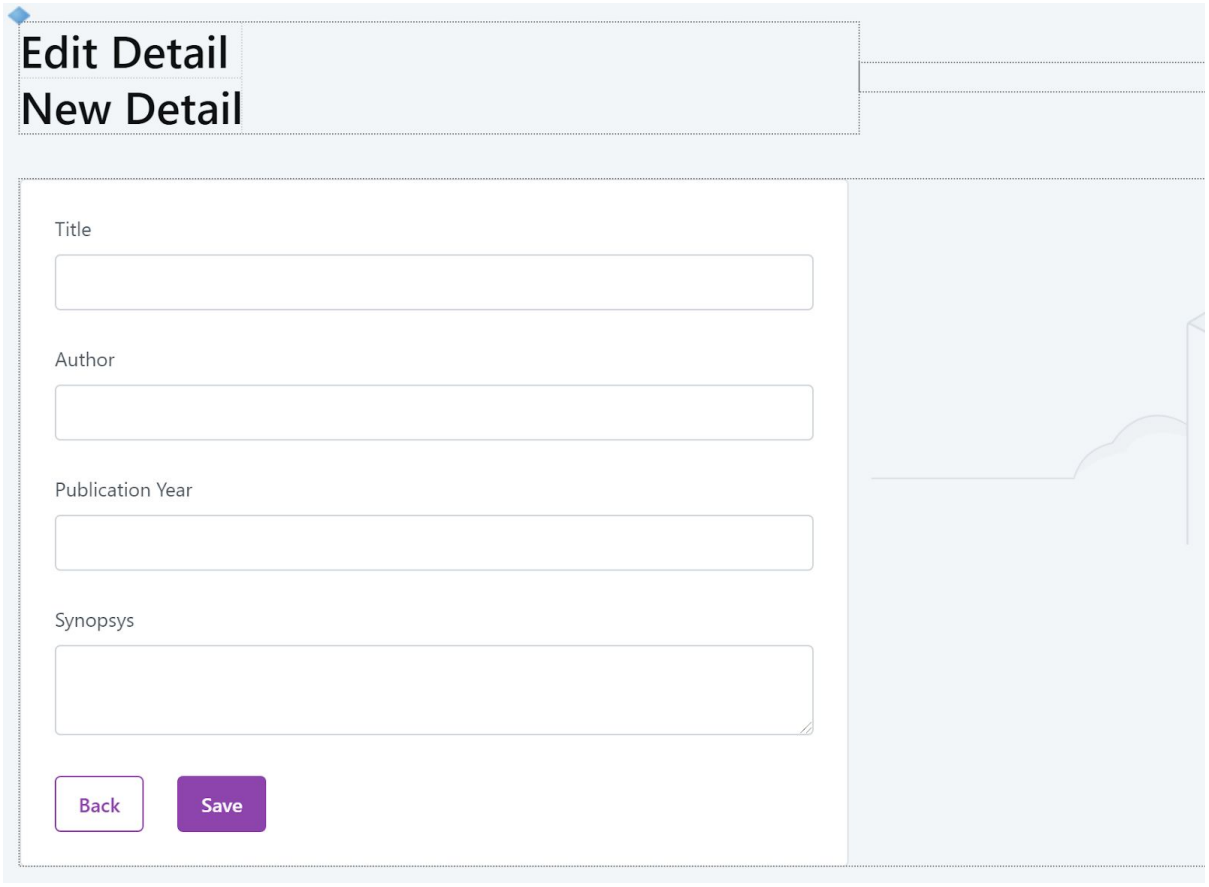
1. Create a new application called **MyLibrary_<*your initials*>** and add a Reactive Module with the same name
2. Create the two entities

a) Start by creating the **Book** entity, with the following attributes:
   - A mandatory *Text* attribute named **Title**, that can hold up to **50** characters
   - A mandatory *Text* attribute named **Author**, with **50** characters length
   - An *Integer* attribute named **PublicationYear**
   - A *Text* attribute named **Synopsis**, that can store **200** characters

b) Then create the **Genre** static entity, with the following entity records:
   - **Novel**
   - **Play**
   - **EpicPoetry** (make sure the record is labeled "**Epic Poetry**")

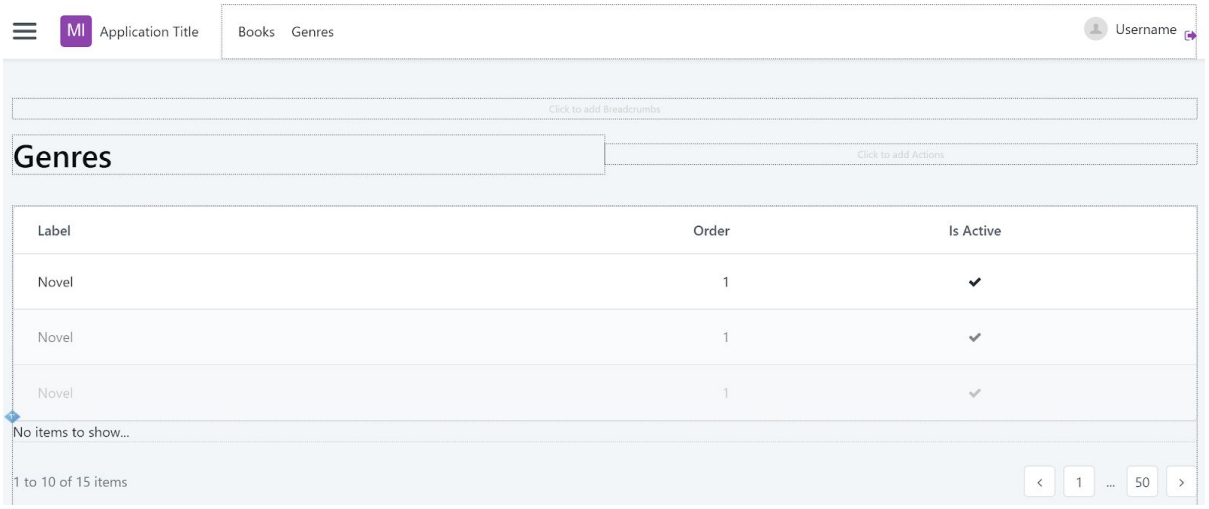
3. Use scaffolding to create a screen that lists all books (without synopsis), connected to another screen that shows the details of a single book. For better usability, make sure that the synopsis uses a multi-line input, and that the menu is labeled **Books**





NOTE: Consider adjusting some of the other names automatically generated to provide a better user experience (like the *Title* placeholder's contents: **Books** instead of *Book List*, **Book** instead of *Detail*)

4. Create another screen that lists all genres. Make sure the menu is labeled **Genres**

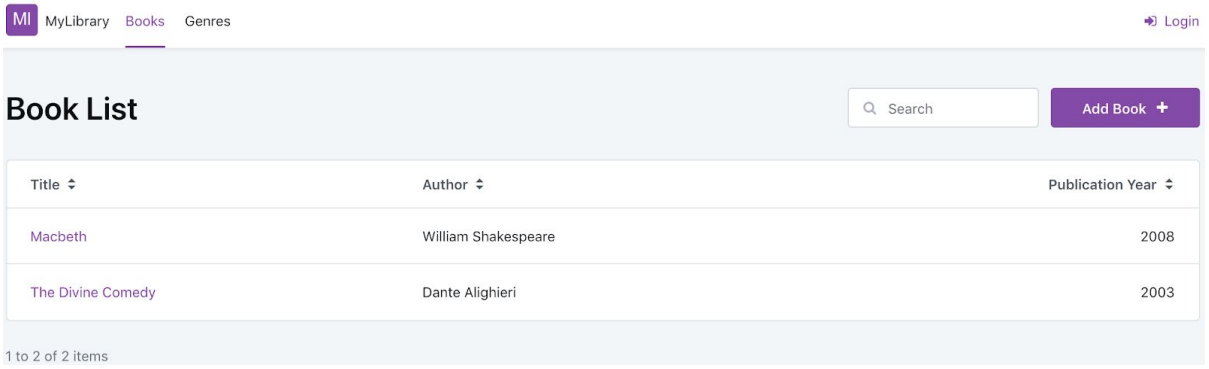


5. Make sure anyone can access those screens
6. Publish your module and use it to add two new books:
    - William Shakespeare's *Macbeth* play, original text version published in 2008, with the following synopsis: *Set mainly in Scotland, the play dramatises the damaging physical and psychological effects of political ambition on those who seek power for its own sake.*
    - Dante Alighieri's *The Divine Comedy* epic poem, published in 2003, with the following synopsis: *The Divine Comedy, is a moving human drama, an unforgettable visionary journey through the infinite torment of Hell, up the arduous slopes of Purgatory, and on to the glorious realm of Paradise - the sphere of universal harmony and eternal salvation.*

QUESTION: Did you manage to type in the full synopsis of *The Divine Comedy*? If not, what stopped you?

CONTINUES NEXT PAGE

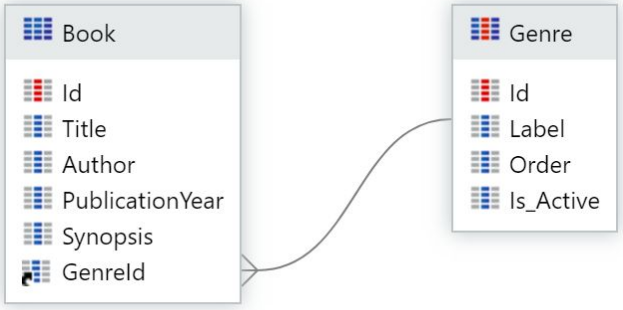
7. Fix the problems identified and make sure all data of both books was correctly inserted.



END OF PART I

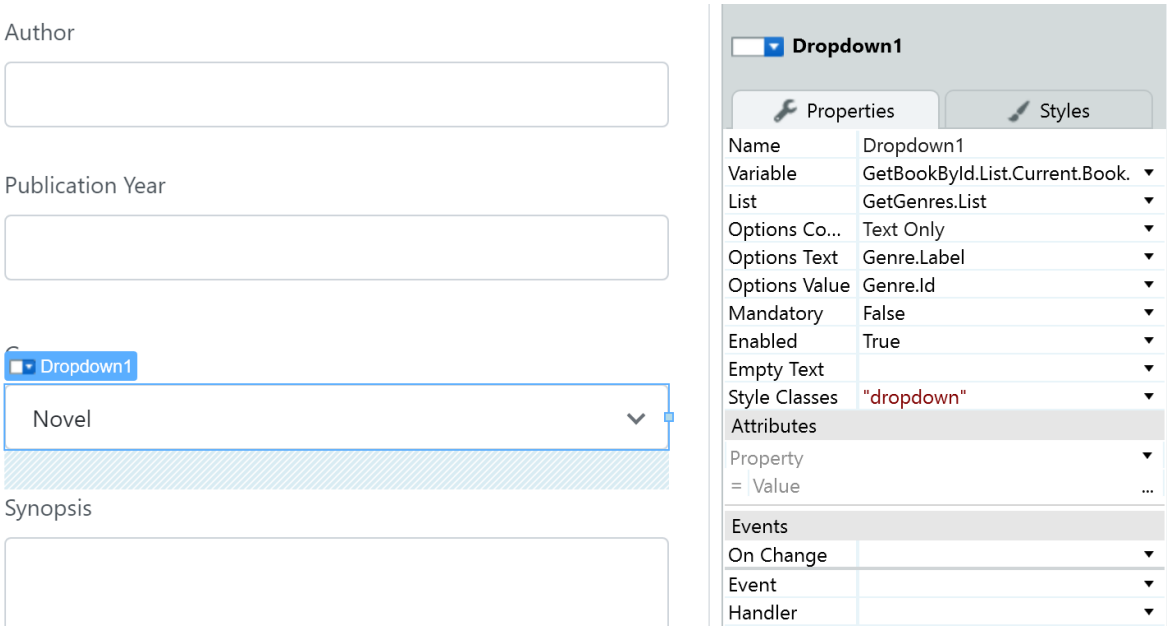
PART II - Extending the data-model with a relationship

1. Establish a relationship between a Book and a Genre



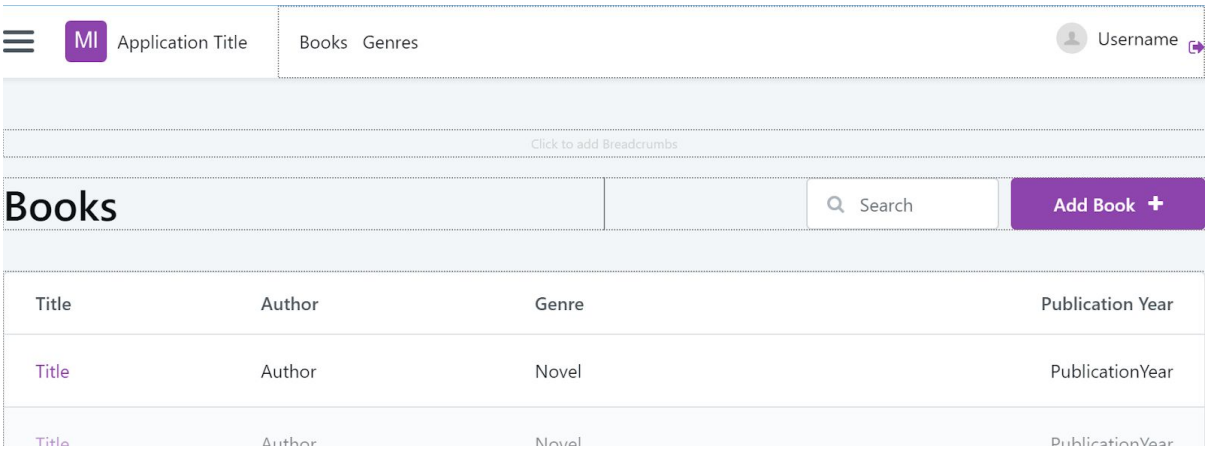
   a) Add a mandatory reference attribute of type *Genre Identifier* to the *Book* entity

2. On the details screen's Form, after the Publication Year, add a mandatory *Dropdown* to choose the genre of the book.



NOTE: Make sure you are using the right variable, the logic to save the record should not require any changes

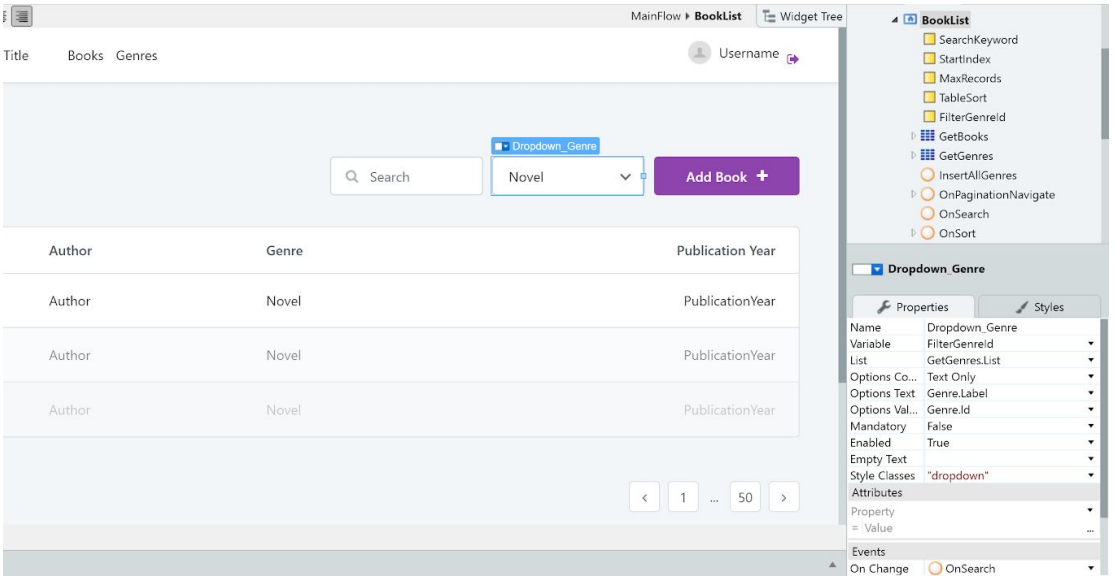
3. On the book listing's screen, add the book's genre to the listing



NOTE: You can drag-and-drop the *GenreId* attribute directly from the *GetBooks* screen aggregate to the Table, and it will automatically add a column labeled **Genre**, displaying the book Genre's *Label*, and also update the aggregate to include the *Genre* source.

When automatically creating Joins, OutSystems uses the Is Mandatory attribute of reference attributes to decide what is the type of Join it creates. Since the GenreId attribute is mandatory, the Join between *Book* and *Genre* defaults to be **Only With**.

4. Still on the book listing's screen, add a dropdown to allow filtering the listing by genre



a) Add a new *Local Variable* of type *Genre Identifier* to store the genre filter

b) Add a new *Aggregate* to the screen to fetch all *Genre* records, sorted by their *Order* attribute

c) Add a *Dropdown* widget to the *Actions* placeholder, between the Search and the Button.
   i. Configure the Dropdown's Variable property with the newly created local variable, and the List property with the result list of the new GetGenres aggregate

   NOTE: This should also automatically configure the *Options Text* and *Options Value* properties, if it doesn't make sure they are set respectively to **Genre.Label** and **Genre.Id**

   ii. Configure the **On Change** event handler with the existing *OnSearch* client action

d) Add a *Filter* to the existing *GetBooks* aggregate to only return Books with the selected genre

5. Publish your module and check that you can filter the books listing by genre and also assign a genre to a book

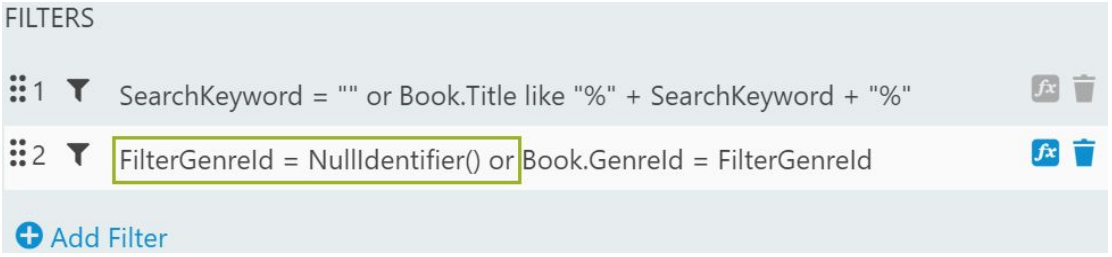
QUESTION: Why can't you see any book?

CONTINUES NEXT PAGE

6. Add an extra option to the dropdown to disable Genre filtering and make it the default
    a) Set the *Empty Text* property of the Dropdown widget to **"(All genres)"**

    NOTE: The *Empty Text* property value will be displayed in the drop down list to represent an Empty selection. When this option is selected, the variable defined will be assigned its default value

    b) Modify the Filter created 4. d) so that when the local variable's value is *NullIdentifier()* it does not filter out any book

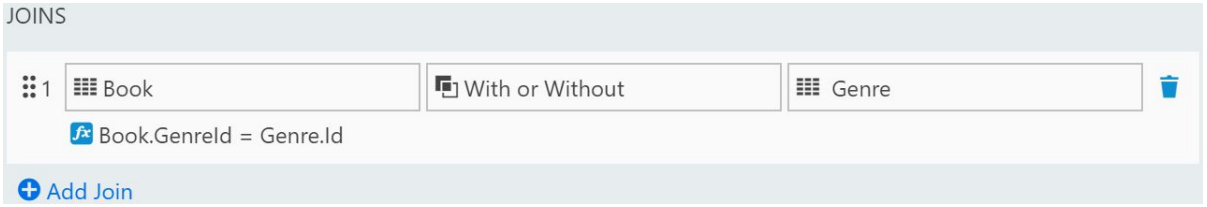


7. Change the *BookDetail* screen so that the Genre dropdown isn't by default set to a valid Genre record
8. Publish your module and check whether you can now filter the books listing by genre and also assign a genre to a book

QUESTION: Why can't you see any book, even when not filtering by genre?

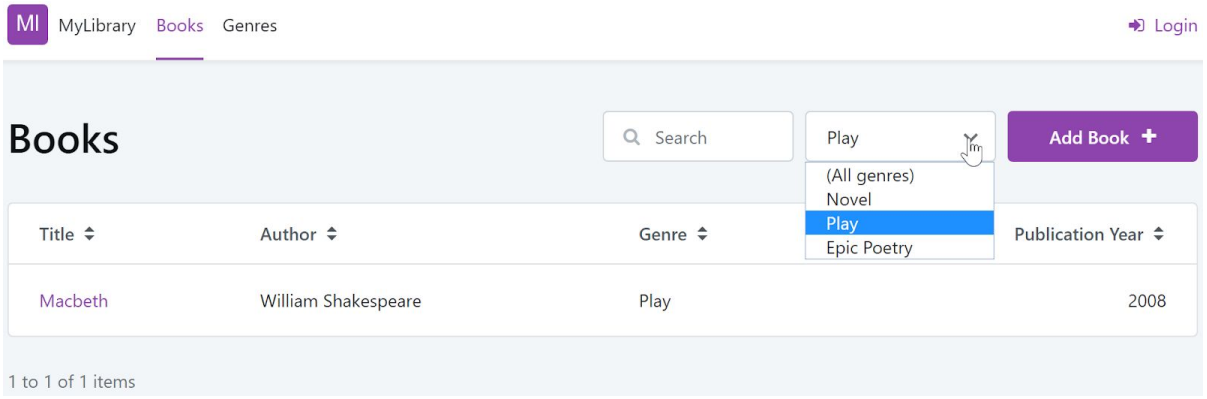
CONTINUES NEXT PAGE

9. In the *GetBooks* aggregate of the *BookList* screen, change the Join type to be **With or Without**



NOTE: This change is only required because there are existing records in the Book entity that do not have the mandatory reference attribute filled.

10. Publish your module and check that you can now see all books unfiltered by default and, after assigning the correct genres to the books, you can now filter the listing.



END OF PART II