Coursework 1 - Report

Ivan ROGER
40285021@napier.ac.uk
Edinburgh Napier University - Advanced Web Technologies (SET09103)

Abstract

This is the my report for the first coursework of the Advanced Web Technologies module (code SET09103) at Edinburgh Napier University.
This coursework consists in the creation of a basic web application to serve a catalogue of entries. This application should be developed using Python Flask.

Keywords – Napier, University, Python, Flask, Web app, Report, MetaChar

1 Introduction

The theme of this coursework being free I had to select mine. Here are my choices.

1.1 Subject and Name

As a choice for my subject I chose to create a catalogue of fictional characters that can appear in movies, comics, tv series and other medias. As a consequence the app is named **MetaChar**, meaning that it is a tool to get some informations and meta-data on characters.

This tools also provides informations on the universe of each character. The main two aspects of this application are the characters and the universes they are from.

1.2 Goals

When I thought of this application I had multiple ideas of what I wish this tools would be able to do.

First of all I wanted to give access to the list of universes present in the database. From this list you should be able to get some details about the universe. And to see all the character that are related to it. Similarly for the characters you can get details and see other characters related to it.

On an other hand i wanted also to be able to search through the entries using a search system. It's implementation is detailed bellow.

At last I tried to give the application a specific graphic style that unified through all the pages.

1.3 Universes

A universe is the data structure that represents the container for a world and a collection of characters.

It is composed of a name, a picture, a short and long description, and a list of medias that have hosted this universe.

1.4 Characters

A character is the data structure used to represent any fictional character that may appear in the universe.

Each character as a name, a picture, a short and a long description, a universe that he belongs to, some tags and other related characters (allies and enemies).

2 Design

This section will focus on detailing the way this application is designed and how the features were implemented.

This figures represents the main pages available.

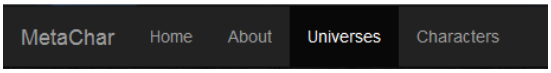

Figure 1: **Navigation** - The navigation links

The first one corresponds to the root of the application. On this page you get general informations on the content of the application. Like the number of entries or other news. (See proposals for enhancements [cf. section 3]).

The second page consists on a page where you can find all the content about the application. For instance information on the creator and links to the documentation and the source code.

The main routes for this application are as following :

/**universes** This route shows the list of universes. (cf. section 2.4)

/**universes/<univID>** This URL gives access to a single universe details (cf. section 2.5)

/**universes/<univID>/characters** Lists all characters in a specific universe.

/**universes/<univID>/characters/<charID>** Similar to the second route this URL gives access to a single character details.

/**characters** Similar to the third route, this one shows the list of all characters regardless of their universe.

The following subsections will explain more deeply how each aspect of the application has been developped.

2.1 Static serving

In order to provide images and other dependancies it has been mandatory to create a specific way to serve files that do not need to be processed on the server.

To do so I used the native static file system provided by *Flask*. Everything to be served staticaly is under a '*static*' directory. Files are ordered by type. Style sheets are under the '*css*' folder, '*img*' contains pictures, '*js*' groups the Javascript needed for Bootstrap as well as '*fonts*'.

2.2 Templates

Each page that can be seen uses a *Jinja* template.

There is a base template that sets up the navigation at the top and the footer of the page.
Each page extends this base template and has a specific content.

Some elements are used on multiple pages. To avoid code duplication I used the macro system that *Jinja* provides. It allows to create a sort of function that returns a specific html content depending on the arguments provided.
Moreover for the character list, two URL use the same template.

Listing 1: The 'data' object

```
1  data = {'config': ..., 'nav': ..., 'active': ..., 'list': ..., 'pages': ..., 'links': ..., 'search': ...}
```

All the data that is needed by the page is passed through a single dictionary called '*data*'.
The '*data*' object has information on the content of the page, the navigation links and the application configuration.

2.3 Storage

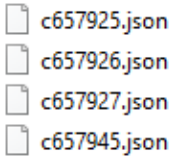

Figure 2: **File storage** - Some files used to store the items

Each item is stored in a plain-text JSON encoded file. All this files are in the '*data*' folder and in a sub-folder depending on whether it is a '*character*' or an '*universe*'.

Folders are automatically scanned at start-up and the minimal content is cached in variables.
The basic data of each item is used the create lists and links to avoid long loading times. When display an entry details we need the full object data so we load the file corresponding to the selected item.

2.4 Entry list

The 'Universes' and 'Characters' pages display a list of the entries that are registered in the application.

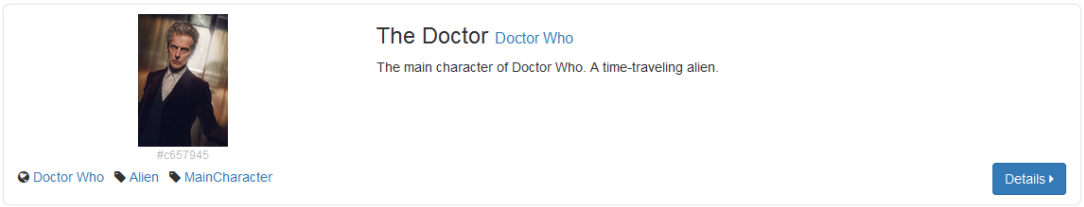

Figure 3: **List entry** - An example of a character entry

Each entry is exposed in a card of its minimal information (that are stored in the cache) via a Jinja macro. These informations consists of the picture of the item and it's name, the associated ID and the short description of the entry.

Some tags are associated to each item and can be used to search through the catalogue (cf. section 2.6).
If the entry is a character we can also see the related universe.

2.5 Item details

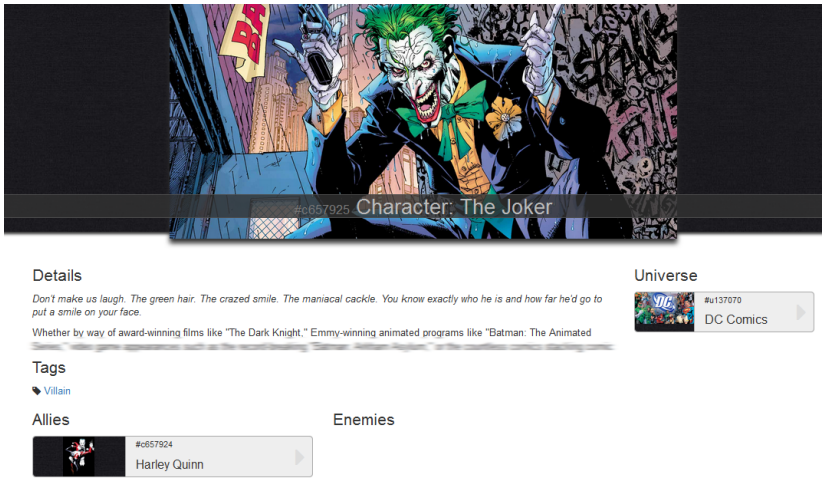

Figure 4: **Entry details** - An example of a character details

Each universe and character can be seen in details and is assigned a unique URL composed of his ID.

On this page we are able to see the picture and name at the top and both short and long descriptions.
Depending on the type of item this page also shows related characters or medias.

2.6 Search system

On the list of entries you can search for a specific entry. On the previous figure we can see that the results are filtered to only show entries of the "Doctor Who" universe and having the "Human" tag. We can also search for text in the name or in the short description.

Figure 5: **Search bar**

Each filter can be combined with others. For example on the figure we use both the tag and universe filter. The last button (the cross) allows the user to clear all filters.

All filters are in the query string so the URL can be bookmarked and re-visited later-on, the content would be exactly the same.

2.7 Pagination

On the catalogue there are many entries and displaying all of them in a single page would make it really long and annoying to read so I created a pagination system to split the content into pages.

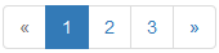

Figure 6: **Pagination system**

The number of items displayed per page can be customized in the configuration.

The navigation keeps the context. So for example, if you were using a filter the next page will show only the content of this same filter.

2.8 Logging

Listing 2: A line of the log file

```
1  INFO | 2016-10-26 11:18:41,173 | main | logRequest | GET: http://localhost:5000/universes
```

The application has a logging system that keeps track of the requests and registers critical steps of processing. Each error is saved in order to be able to ensure that the application is working properly at all times.

2.9 Graphic style

A consequent amount of time has been spent on trying to make the application look nice and being user-friendly. I tried my best to make sure that everything was intuitive and easy to use.

I have a followed some common guides-lines through the pages and did my best to keep the same colour scheme and to have the items ordered the same way on all pages .

3 Enhancements

Some of these enhancements are available on github in the issue tracker as features.

A) Creation and Edition One of the first things that should be implemented if this project was to be really used would be to provide the ability to create new entries in the catalogue and the possibility to edit them afterwards.

B) Featured items on the home page The home page is currently quite empty but could show a list of the last items that were added, the one that are the most viewed etc ...

C) More user-friendly search system It is currently not as easy to use the search system that I wanted it to be. For example we could provide an auto-complete search for the tags (in Javascript).

D) Graphic style re-use In the universes or characters list we should re-use the graphic style of the links in the details page.

E) Optimize load time When displaying a list all entries are processed but only 5 at most are shown by page. To win some loading time we could process only the entries that would be displayed in the end.

F) Accounts and deletion It is possible that we may want to delete an entry. To do so it would be necessary to create an account system to only allow this critical action to administrators.

G) Unit tests To make sure that the app always works when used in production we should redact en run some unit tests. So that the application is in a continuous integration.

H) Medias details The current system to show media related to a universe is not really welcoming. To make it more friendly we could shows those information on a separate page and provide more details on each media, like the name, author(s) and release dates.

I) Translations Use different templates depending on the user locale variable. Using babel maybe ?

4 Evaluation

In this section we will make a critical self-assessment of the result and current state of this project.

4.1 Advantages

I consider that the main advantage of this application is its good appearance. As described in the section 2.9 I really did my best to have a good user interface. The few features that are implemented are fully finished. I preferred to make sure that the thing I had done worked perfectly well before done anything else.

4.2 Disadvantages

On another hand this application remains quite basic and could provide a lot more features. As the Enhancements sections shows there is a lot more that could have been done.

If I had more time and managed it's use more wisely. I probably spent too much time trying to get the app look good but which gave me less time to implement new functionalities.

5 Conclusion

5.1 Difficulties

I had some troubles to manage my time for this coursework because I didn't plan exactly what I was going to do so I ended up doing most of the stuff during the last week and still hadn't redacted my report.

I spent another amount of time on the search system and the pagination. As I didn't really know how and what I exactly wanted to have.

The rules for this coursework being a bit vague I had to take some decision and judge by myself which choice were good and which weren't.

5.2 Things I learned

This project gave me confidence in my ability to start and create an application from scratch. I have learned from the difficulties I encountered.

I learned that it is important to redact what I am going to do before beginning to do anything and if needed modify expectation during the development.
I learned how to use Flask and strengthened my skills using python as a programming language.
I had the chance to think and design a URL pseudo-API by myself and to implement it.
This project was also a place for me to use my self-learning skills as I had to face some errors and couldn't ask every single time I had a problem.

6 Ressources

Data and informations

- DC Comics - Official website : Character list
- Wikipedia - Details on characters
- Tardis Wikia : Details on characters

Code and frameworks

- Bootstrap : CSS framework
- Font Awesome : Icon font pack
- Markdown (for python)

Documentation

- Jinja2
- Flask
- Werkzeug (python framework used by Flask)

Report

- Coursework report template