KOD

GUIDA INTERGALATTICA PER SCAPISTI

CAPITOLO 1 - REV 1

Intro

Prima di cominciare:

- Un code editor come Visual Studio Code, Atom, Notepad++, Ecc.
- Aprite il vostro Browser preferito al seguente indirizzo **https://regex101.com/**
- Attivare il logging di debug dalle impostazioni di KoD in: Generale > Abilita logging di debug
- Tanta pazienza :p

Creazione di un canale

Per la creazione di un canale per KOD (ma anche per Alfa) abbiamo bisogno di 2 files:

- nomecanale.json (Questo file serve per le impostazioni base del canale)
- nomecanale.py (Questo è il vero cuore del canale)

Per questa guida ho scelto di utilizzare il sito **Cinema Streaming**, questo sito offre molte informazioni sui film ma presenta anche alcune difficoltà che impareremo (forse) ad aggirare.

Si comincia

Creiamo quindi il primo file di cui abbiamo bisogno e nominiamolo **cinemastreaming.json**

```
01  {
02      "id": "cinemastreaming",
03      "name": "Cinema Streaming",
04      "language": ["ita"],
05      "active": true,
06      "adult": false,
07      "thumbnail": "https://cinemastreaming.info/wp-content/uploads/2017/10/CinemaStreaming300dpi.png",
08      "banner": "https://cinemastreaming.info/wp-content/uploads/2017/10/CinemaStreaming300dpi.png",
09      "categories": ["tvshow", "movie", "anime"]
10  }
```

Questo codice non ha bisogno di molte spiegazioni, in ogni caso:

- id = Nome del file .py a cui fa riferimento
- name = È il nome del canale che apparirà nei menu
- active = (true / false) indica se il canale è attivo o meno , se false il canale viene nascosto
- adult = (true / false) indica se contiene o meno contenuti per adulti
- thumbnail e banner = Sono i link alle rispettive immagini
- categories = Indica in quali categorie viene inserito il canale, le categorie sono: "movie", "tvshow", "anime", "documentary", ecc.

Passiamo ora al secondo file e nominiamolo **cinemastreaming.py**

Prima di iniziare con il file .py una piccola e doverosa precisazione:

A differenza di altri linguaggi che delimitano blocchi di codice con parentesi graffe Python si è scelto di usare l'indentazione ossia un rientro, quindi assicuratevi di utilizzare una sola tipologia di indentazione in tutto il vostro file!

Qualsiasi code editor permette di settare i parametri di indentazione, io utilizzo 4 spazi.

Inseriamo questa come prima riga del codice (ne indica la codifica)

```
001 # -*- coding: utf-8 -*-
```

Continuiamo con alcune righe prettamente informative (ogni riga preceduta da # non viene interpretata, in gergo si dice "commentata")

```
002 # ------------------------------------------------------------
003 # Canale per cinemastreaming
004 # ------------------------------------------------------------
```

Importiamo ora i moduli di cui avremmo sicuramente bisogno per lo scaping del canale
(i moduli sono file o pacchetti di file .py contenenti delle funzioni)

```
005 import re
006
007 from channels import filtertools
008 from core import scrapertools, servertools, httptools
009 from core.item import Item
010 from platformcode import config
```

Per ora non mi soffermo sul loro funzionamento sappiate comunque che l'importazione funziona in questa maniera:

from **cartella** import **modulo**

Come notate nella riga **10** vediamo un core.item richiama il file item.pt dalla cartella core mentre import importa un'unica funzione del suddetto file.

Continuiamo creando una variabile (host) a cui diamo come valore il link al sito

```
011 host = 'https://cinemastreaming.info'
```

Aggiungiamo subito dopo una seconda variabile

```
012 headers = [['Referer', host]]
```

Questa variabile potrebbe servire in caso il sito in questione non dia acceso ai video se non c'è un riferimento al sito da cui provengono

Ora creiamo il menu principale che vediamo all'apertura di un canale

```
013 def mainlist(item):
014     itemlist = [Item(channel = item.channel,
015                      contentType = 'movie',
016                      title = 'Film',
017                      url = host + '/film/',
018                      action = 'peliculas',
019                      thumbnail = '',
020                      fanart = ''
021                      ),
022     ]
023     return itemlist
```

Brevemente

- def = definisce una funzione
- mainlist = Nome della funzione (in questo caso la funzione deve obbligatoriamente avere questo nome)
- (item) = Il parametro della funzione
- itemlist = Lista di parametri
- Item = Insieme di parametri

Per la variabile item:

- channel = Lasciate item.channel
- contentType = Tipo di contenuto
- url = Link alla pagina a cui si vuole arrivare, in questo caso host + '/film/' ossia la variabile che abbiamo creato prima a cui aggiungiamo (concateniamo) la stringa della sottopagina del sito
- action = La funzione da avviare
- thumbnail, fanart = Eventuali link ad immagini

Aggiungiamo a quasta funzione un log per un eventuale debug, per fare questo importiamo una nuova funzione da un modulo di KOD

```
024 from channels.support import log
```

Questo è una funzione semplificata per il log che ci permette di avere un log scrivendo meno testo (sono pigro :p)

“Il file support.py permette di semplificare molto lo scraping dei siti, ma di questo ne parleremo approfonditamente in un altro capitolo.”

Aggiungiamo quindi alla funziona mainlist

```
025 def mainlist(item):
026     log()
027     itemlist = [Item(channel = item.channel,
028                      contentType = 'movie',
029                      title = 'Film',
030                      url = host + '/film/',
031                      action = 'peliculas',
032                      thumbnail = '',
033                      fanart = ''
034                      ),
035     ]
036     return itemlist
```

Apriamo ora, da KOD, il canale creato.

Dal log di Kodi vedremo la seguente stringa

```
01   [cinemastreaming.py] - [mainlist]
```

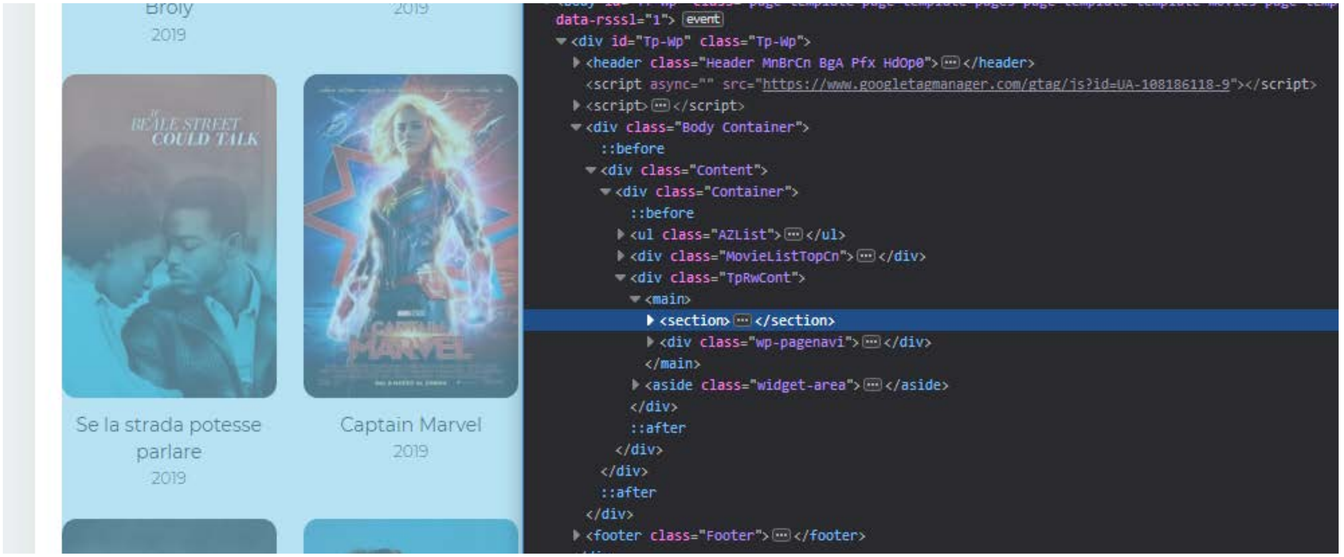
Passiamo ora alla prima funzione di scaping ossia trovare nella pagina i film presenti con le varie informazioni
Creiamo dunque la seguente funzione:

```
037 def peliculas(item):
038     log()
039     itemlist = []
040     data = httptools.downloadpage(item.url, headers=headers).data
```

comandi utilizzati sono:

- itemlist = [] Crea una lista vuota
- Mentre il comando successivo scarica la pagina e la mette nella variabile data

Apriamo ora la pagina **https://cinemastreaming.fun/film/** clicchiamo col destro nella zona dove sono elencate le varie locandine.



Dovremmo vedere qualcosa simile a questo.

La parte che ci interessa è quella compresa fra <main> e </main>, ossia il blocco che comprende i film e i pulsanti di paginazione.

Utilizziamo quindi il seguente comando per "isolare questo blocco"

```
041     block = scrapertools.find_single_match(data, r'<main>(.*?)<\/main>')
```

Quindi:

- Invochiamo la funzione **find_single_match** (trova corrispondenza) del modulo **scrapertools** nella variabile **data** di tutto ciò che è compreso fra <main> e <\/main> e inseriamola in una variabile, in questo caso **block**

Vedremo fra poco il significato di **(.*?)**

Togliamo dal nostro blocco tabulazioni e interruzioni di riga con:

```
042 block = re.sub('\t|\n', '', block)
```

Ovvero:

- \t = Tabulazione
- \n = Interruzione di riga
- | = Oppure
- '' = Sostituiamoli con nulla

Aggiungiamo quindi un altro log

```
043    log('block= ',block)
```

Proviamo ora ad aprire Film dal menu di Cinema Streaming, controlliamo quindi il log e copiamo la riga corrispondente.

```
01   [log] [cinemastreaming.py] - [peliculas] block= <section><div class="Top"><h1 class="Title">Film<...
```

Andiamo sul sito **https://regex101.com/** e incolliamola nel riquadro **TEST STRING**

Regex, questo sconosciuto

Siamo arrivati alla parte in cui dobbiamo ricavare le informazioni dalla nostra pagina, qui ci vengono in aiuto le **Espressioni Regolari** o **Regex**

Il sito **Regex101** è molto semplice da utilizzare:

- In alto abbiamo l'area **REGULAR EXPRESSION** qui è dove testeremo il nostro regex
- Subito sotto abbiamo l'area **TEST STRING** dove incolleremo il nostro codice
- A destra, in alto abbiamo l'area **EXPLANATION** ci verranno dati suggerimenti man mano che scriviamo
- A destra, in centro abbiamo le **MATCH INFORMATION** ovvero tutti i riscontri che troveremo nella pagina
- A destra, in basso ci sono le **QUICK REFERENCE**, una piccola guida

Le informazioni che cerchiamo sono comprese fra <li class="TPostMv"> e

Le espressioni regolari più comuni sono:

- .*? = Qualsiasi carattere fino a (il carattere o insieme di caratteri successivi es: .*?<a href=")
- [^"]+ = Simile al primo ma non permette che sia nullo (es: [^<]+ Seleziona tutti i caratteri fino a <)
- [a-z] = Solo lettere minuscole (seguito da un + Si ripete fino ad un carattere diverso)
- [A-Z] = Solo lettere maiuscole (seguito da un + Si ripete fino ad un carattere diverso)
- [0-9] o \d= Solo numeri (seguito da un + Si ripete fino ad un carattere diverso)
- \D = Tutto ciò che non è un numero
- \w = Numeri o lettere
- \W = Tutto ciò che non è ne numero ne lettera

NOTA BENE se troviamo il simbolo "**/** " dobbiamo farlo precedere da "****" altrimenti restituisce errore

Se mettiamo fra parentesi le espressioni sopra indicate "catturiamo" quella parte di stringa.

Vediamo quindi come catturare le informazioni che ci servono:

```
044     patron = r'<article.*?class="TPost C">.*?<a href="([^"]+)">.*?src="([^"]+)".*?>.*?<h3 class="Title">([^<]+)<\/h3>(.*?)<\/article>'
```

Con questo regex catturiamo in ordine:

- URL
- Immagine
- Titolo
- E un blocco che contiene altre informazioni come anno, regista ecc...

Continuiamo col comando

```
045    matches = re.compile(patron, re.DOTALL).findall(block)
```

Che trova tutte le corrispondenze del **patron** nella stringa **block** e la mette in una lista **matches**

Continuiamo con

```
046    for scrapedurl, scrapedthumb, scrapedtitle, scrapedinfo in matches:
```

Tradotto, assegna a delle variabili (scrapedurl, scrapedthum, ecc) ognuno dei 4 elementi catturati dal comando precedente

Possiamo vederne il risultato con il log (indentato rispetto a for) per esempio:

```
047        log('Titolo = ', scrapedtitle
```

E nel nostro log appariranno tutti i titoli

```
01   [log] [cinemastreaming.py] - [peliculas] Titolo = The Prodigy &#8211; Il figlio del male
02   [log] [cinemastreaming.py] - [peliculas] Titolo = Star Wars: Episodio IX
03   [log] [cinemastreaming.py] - [peliculas] Titolo = Artemis Fowl
04   [log] [cinemastreaming.py] - [peliculas] Titolo = Aladdin (2019)
05   [log] [cinemastreaming.py] - [peliculas] Titolo = Dragon Ball Super: Broly
06   [log] [cinemastreaming.py] - [peliculas] Titolo = The vanishing &#8211; Il mistero del faro
07   [log] [cinemastreaming.py] - [peliculas] Titolo = Ancora auguri per la tua morte
08   [log] [cinemastreaming.py] - [peliculas] Titolo = The Front Runner &#8211; Il vizio del potere
```

```
09  ...
```

Quello che ci serve ora però è il il 4° Valore "**scrapedinfo**"

Quindi modifichiamo il nostro log in questo modo:

```
048        log('info = ', scrapedinfo)
```

E copiamo uno qualsiasi dei risultati del log, passiamo per regex101 e troviamo il seguente regex:

```
049        patron = r'<span class="Year">(.*?)<\/span>.*?<span class="Vote.*?">(.*?)<\/span>.*?<div class="Description"><p>(.*?)<\/p>.*?<p class="Genre.*?">(.*?)<\/p><p class="Director.*?">.*?<a.*?>(.*?)<\/a>.*?<p clas-
    s="Actors.*?">(.*?)<\/p>'
```

Continuiamo con le solite istruzioni:

```
050         info = re.compile(patron, re.DOTALL).findall(scrapedinfo)
051         for year, rating, plot, genre, director, cast in info:
```

In questo caso per le variabili genre e cast abbiamo più risultati, quindi indentiamo i seguenti comandi:

```
052             genre = scrapertools.find_multiple_matches(genre, r'<a.*?>(.*?)<\/a>')
053             cast = scrapertools.find_multiple_matches(cast, r'<a.*?>(.*?)<\/a>')
```

In questo caso la funzione **find_multiple_matches** trova tutti i risultati del patron

Continuiamo inserendo questi risultati nei **infoLabels** cioè quelle informazioni aggiuntive che appaiono sul nostro amato Kodi alla pressione dal taso info

```
054             infoLabels = {}
055             infoLabels['Year'] = year
056             infoLabels['Rating'] = rating
057             infoLabels['Plot'] = plot
058             infoLabels['Genre'] = genre
059             infoLabels['Director'] = director
060             infoLabels['Cast'] = cast
```

Creiamo prima una variabile vuota e inseriamo uno di seguito all'altro tutte le informazioni che abbiamo ottenuto.

Inseriamo ora tutte le informazioni in **itemlist**

```
061             itemlist.append(
062                 Item(channel=item.channel,
063                      action="findvideos",
064                      contentType=item.contentType,
065                      title=scrapedtitle,
066                      fulltitle=scrapedtitle,
067                      url=scrapedurl,
068                      thumbnail=scrapedthumb,
069                      infoLabels = infoLabels,
070                      show=scrapedtitle,))
```

Oltre ai valori che abbiamo visto prima ci soffermiamo su:

- title = Il titolo visualizzato
- fulltitle = Il titolo che viene passato a imdb
- Show = Il titolo che compare quando imdb, non trovandolo nel database chiede di modificare

Infine visualizziamo itemlist con:

```
071    return itemlist
```

Torniamo a KOD e vediamo che ora vengono visualizzati tutti i titoli e le informazioni dei film trovati, ma solo della prima pagina,

Aggiungiamo quindi, sempre aiutandoci con regex101

Il seguente codice, subito prima di **return itemlist**

```
072     patron = '<a class="next page-numbers" href="([^"]+)">'
073     next_page = scrapertools.find_single_match(data, patron)
074
075     if next_page != "":
076         itemlist.append(
077             Item(channel=item.channel,
078                 action="peliculas",
079                 title="[B]" + config.get_localized_string(30992) + "[/B]",
080                 url=next_page))
```

config.get_localized_string(30992) = Richiama la stringa 30992 nei file di traduzione e corrisponde a “Pagina Successiva”

action rimanda alla stessa funzione peliculas ma con url trovata tramite patron e scrapedtools

Findvideos: guardiamoci un film

Siamo arrivati alla definizione della funzione clou, quella che ci permetterà di trovare i server e riprodurre finalmente il video.

Cominciamo con

```
081  def findvideos(item):
082     log()
083
084     data = httptools.downloadpage(item.url, headers=headers).data
085     log(“download page= “,data)
```

Cinema Streaming ha il brutto vizio di inserire nella lita dei film anche le anteprime, senza però inserire i link ai server, selezioniamo quindi un video di cui siamo sicuri di avere i server, copiamo il nostro log e incolliamolo su regex101, aiutandoci anche col browser (come al solito tasto destro > Analizza elemento)

Scopriamo che il video viene riprodotto in un iframe, quindi:

```
086     patron = r'<div class=”TPlayerTb.*?src=(?:”|&amp;quot;)(.*?)(?:”|&amp;quot;)'
087     matches = scrapertools.find_multiple_matches(data, patron)
088     log('matches = ',matches)
```

Andando a provare i link nel log, ci accorgiamo, purtroppo che ci riportano all'homepage.

Come risolvere , quindi questo ennesimo problema?

Qui basterebbe conoscere un po' di html per capirlo, ma sono buono e vi do un aiuto :p

Nei link che vedete compaiono “&” e “&” entrambi corrispondono a... “&”

basteà quindi sostituirli aggiungendo subito dopo data = Il seguente comando

```
089    data = re.sub('#038;|amp;','',data)
```

Che significa trova #038; oppure amp; in data e sostituiscili con nulla.

A questo punto basterebbe dare in pasto a servertools i risultati ottenuti, ma non in questo caso dato che l'iframe contiene in secondo iframe con il collegamento al video, passiamo allora i risultati ottenuti a

```
090     urls = ''
091     for match in matches:
092         urls += httptools.downloadpage(match, headers=headers).data
```

Abbiamo creato così una variabile (urls) vuota, scaricato la pagina per ogni risultato trovato e accodate nella variabile urls (con il segno + Prima di uguale)

E passiamo tutto a

```
093     itemlist = servertools.find_video_items(data=urls)
```

Facciamo fare a servertools il lavoro sporco e godiamoci i risultati con

```
094    for videoitem in itemlist:
095         videoitem.title = item.title + '[COLOR green][B]' + videoitem.title + '[/B][/COLOR]'
096         videoitem.fulltitle = item.fulltitle
097         videoitem.show = item.show
098         videoitem.thumbnail = item.thumbnail
099         videoitem.channel = item.channel
100         videoitem.contentType = item.contentType
101
102     return itemlist
```

Passando cioè ai vari videoitem i valori che vogliamo

Se avete fatto tutto potete finalmente godervi il vostro film!

Eliminate o commentate ora tutti i log superflui.

P.S. Per eliminare i video senza link c'è una soluzione...

Vediamo chi riesce a intuirla :p !

Continua...