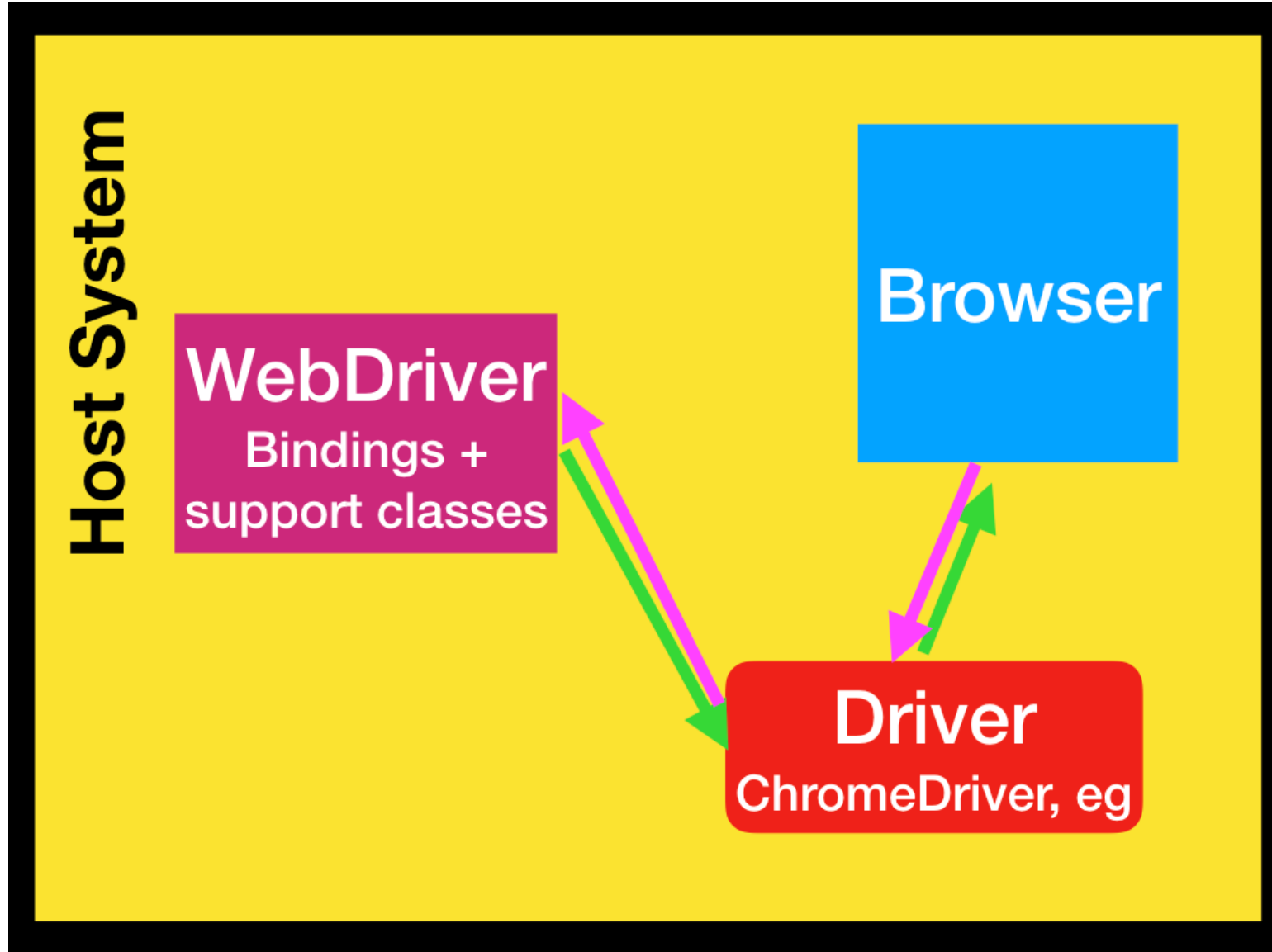

Selenium

Selenium

- Selenium è un tool *open source* per la gestione automatizzata dei browser, utilizzato come framework di testing.
- Selenium è in realtà una suite, composta da diversi strumenti:
 - Selenium IDE,
 - Selenium Builder,
 - Selenium Grid ,
 - Selenium WebDriver.

WebDriver



Selenium.WebDriver

- Selenium WebDriver (successore di Selenium Remote Control - Selenium RC) è uno strumento che simula il comportamento di un utente reale all'interno di un browser:
 - viene utilizzato per eseguire localmente o su macchine remote i test all'interno dei browser supportati
 - tutti i principali, come Firefox, Safari, Edge, Chrome, Internet Explorer...

Anaconda


- La libreria Selenium.WebDriver viene fornita nei principali linguaggi di programmazione
- Per Anaconda usare le indicazioni alla pagina:
<https://anaconda.org/conda-forge/selenium>


```
conda install -c conda-forge selenium
```


Chromedriver


- Il *chromedriver* è un programma che permette di creare il ponte tra la libreria Selenium e il browser web Chrome
- Verificare la versione di Chrome:

Informazioni su Chrome

 Google Chrome









 Chrome è aggiornato
Versione 108.0.5359.125 (Build ufficiale) (a 64 bit)

Ricevi assistenza per Chrome 

Segnala un problema 

Chromedriver

- Scaricare dal sito:
<https://chromedriver.storage.googleapis.com/index.html>
- La versione relativa (ver. chrome <= ver. chrdrv)
- Nel nostro caso la 109.*.71

	107.0.5304.18	-
	107.0.5304.62	-
	108.0.5359.22	-
	108.0.5359.71	-
	109.0.5414.25	-
	icons	-
	LATEST_RELEASE	2022-11-30 05:17:18
	LATEST_RELEASE_100	2022-03-30 07:06:45

Driver

- Copiare il contenuto del file zip (*chromedriver.exe*) nella directory del progetto
- In ogni directory di progetto copieremo una versione del programma di driver per motivi di compatibilità
- Ovviamente il browser Chrome deve essere preinstallato
- Esistono driver anche per
 - Edge (<https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/>)
 - Firefox (<https://github.com/mozilla/geckodriver/releases>)
 - Safari (non ne ha bisogno ha già un driver interno)

Case Study

- Per imparare ad usare Selenium la cosa migliore è applicare un case study
- Esempio: imdb.com: paginazione e spoiler

The screenshot shows a movie review on IMDb. At the top, there is a star rating of 10/10. Below it, the review text reads: "This film will always rightly stand out as one of the most iconic films of the last nearly 30 years." followed by the reviewer's name "timlee19" and the date "8 April 2021". A red oval highlights the text "Warning: Spoilers" in red. Below this, there is a helpfulness section that says "0 out of 0 found this helpful. Was this review helpful?" with "Yes" and "No" buttons. A red arrow points from the "Warning: Spoilers" text to a red oval containing a blue downward-pointing chevron icon, which is a common UI element for expanding or collapsing content. Below the helpfulness section, the review text continues: "Demme shows that you don't need a huge budget to make a great film." and "What the film doesn't show you is as powerful (in some ways more so) than what it does, and this is one of the things that it does well, and which it's sequel (2001's 'Hannibal') lacked in spades. It allows you, the viewer to fill in the gaps; giving you a peep into a world of depraved madness and letting you imagine the rest. Which makes it in turn such an effectively scary film."

★ 10/10

This film will always rightly stand out as one of the most iconic films of the last nearly 30 years.

timlee19 8 April 2021

Warning: Spoilers

0 out of 0 found this helpful. Was this review helpful? Yes No

[Report this](#) | [Permalink](#)

Demme shows that you don't need a huge budget to make a great film.

What the film doesn't show you is as powerful (in some ways more so) than what it does, and this is one of the things that it does well, and which it's sequel (2001's 'Hannibal') lacked in spades. It allows you, the viewer to fill in the gaps; giving you a peep into a world of depraved madness and letting you imagine the rest. Which makes it in turn such an effectively scary film.

JavaScript

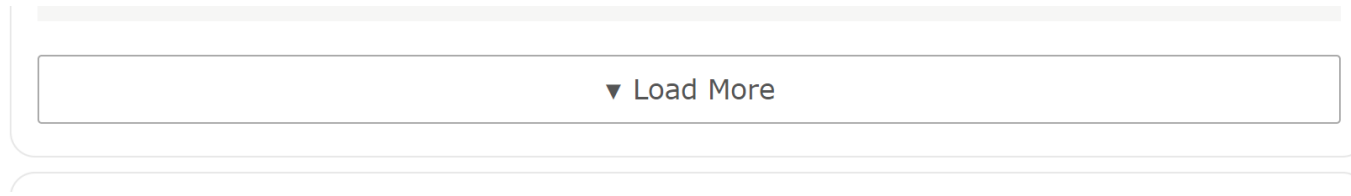
- Cosa è successo?
- Se apriamo la pagina di imdb all'indirizzo https://www.imdb.com/title/tt0042332/reviews?ref_=t_ov_rt possiamo leggere le recensioni che utenti *imdb* hanno postato sul sito a riguardo del famoso film d'animazione della Walt Disney *Cenerentola* (del 1950)
- Il sito pubblica le recensioni facendo attenzione che nel testo non ci siano riferimenti troppo espliciti sulla trama
- Nel caso, non carica la recensione nella pagina, avverte il lettore con il messaggio **Warning: Spoiler** e posiziona un bottone in basso a destra per scaricare *volontariamente* la recensione

JavaScript

- Al *click* del mouse sulla freccetta di espansione il browser web esegue una nuova connessione con il server *imdb* e scaricare la recensione *censurata*
- A questo punto il browser mostra la recensione
- Questa attività richiede necessariamente che l'utente esegua il *click fisicamente*
- La libreria `requests` di python non è capace di eseguire questa attività in quanto essa si limita a scaricare l'HTML di partenza e non può, ovviamente, interagire con i *javascript* della pagina

IMDB

- Inoltre in fondo alla pagina troviamo:



- Gestione della paginazione delle recensioni
- Anche in questo caso il click sul bottone ***Load More*** farà eseguire a Chrome uno script js per scaricare nuove recensioni (25 alla volta)
- Una volta scaricate tutte le recensioni il bottone scompare

Selenium

- Qui entra in gioco Selenium
- Sebbene creato per altri scopi torna utile per permettere ad un *webscraper* di eseguire un *click* su un bottone in una pagina web da programma
- Molte azioni *umane* possono essere simulate
- Esempio facile:
 - Aprire la pagina: <https://www.imdb.com/>
 - Scrivere nella casella di testo il titolo Cenerentola
 - Cliccare sulla lente di ingrandimento

Selenium

- Funzione per la creazione di un canale di comunicazione con Chrome (utile anche in seguito)

```
def start_driver():  
    chrome_options = webdriver.ChromeOptions()  
    # La directory deve essere creata  
    chrome_options.add_argument(  
        r'user-data-dir=C:\Users\massi\Google\Data')  
    # Profilo di default  
    chrome_options.add_argument(  
        r'--profile-directory=Default')  
    # lancio del driver, che lancerà Chrome  
    ser = Service("chromedriver.exe")  
  
    driver = webdriver.Chrome(  
        service=ser,  
        options=chrome_options)  
  
    return driver
```

Selenium

```
driver = start_driver()  # apre il browser
driver.get("https://www.imdb.com")  # apre la pagina

# trova gli elementi html con cui interagire
text = driver.find_element_by_id("suggestion-search")
button = driver.find_element(By.ID, "suggestion-search-button")

# simula la scrittura della casella di testo
text.send_keys("Cenerentola")
# simula il click del mouse sul bottone
button.click()
```

Funzioni utili

- Driver
 - `get(url)`
 - `quit()`
 - `find_element[s](By.*, TAG)`
 - `dove * =`
 - `id`
 - `xpath`
 - `link_text`
 - `name`
 - `tag_name`
 - `class_name`
 - `css_selector`
 - l'opzione `s` ritorna tutti gli elementi
 - `page_source`
- WebElement
 - `send_keys()`
 - `click()`
 - `submit()`
 - `get_attribute(attr)`

Funzioni utili - 2

- Driver
 - By
 - ID = "id"
 - XPATH = "xpath"
 - LINK_TEXT = "link text"
 - PARTIAL_LINK_TEXT = "partial link text"
 - NAME = "name"
 - TAG_NAME = "tag name"
 - CLASS_NAME = "class name"
 - CSS_SELECTOR = "css selector"

```
text = driver.find_element(By.ID, "suggestion-search")
```

Funzioni utili - 3

- WebDriver
 - `execute_script (script)`
 - `get_cookies ()`
 - `refresh ()`
- WebElement
 - `is_[displayed() | selected() | enabled()]`

Selenium + BeautifulSoup

- Spesso le due librerie sono in grado di collaborare:
 - Selenium raggiunge l'HTML utile
 - BeautifulSoup lo analizza

IMDB

- Passiamo ora all'analisi del problema relativo al case study
- Dobbiamo
 - Caricare la pagina
 - Scaricare tutte le recensioni cliccando più volte sul bottone *Load More*
 - Localizzare i bottoni di espansione
 - Cliccare su tutti i bottoni di espansione

Load More

- Il bottone *Load More*
- È collocato alla fine della lista delle reviews
- Ad ogni click ne carica altre 25 per volta
- Dopo aver caricato tutte le review il bottone non appare (not displayed)
- Dopo una ispezione si ottiene:

```
<button class="ipl-load-more__button" data-target-  
container="reviews-container" id="load-more-trigger">Load  
More</button>
```

Load More

```
load_more = driver.find_element(By.ID, "load-more-trigger")
while load_more.is_displayed():
    // time.sleep(1)
    load_more.click()
    # no d.o.s. but also for waiting for data
    // time.sleep(2)
    # potrebbe essere cambiato l'element, quindi lo cerco
    # di nuovo
    load_more = driver.find_element(By.ID, "load-more-trigger")
```

Spoiler

- A questo punto tutte le reviews sono state scaricate
- Dobbiamo adesso espandere le reviews di tipo *spoiler* usando i bottoni di espansione
- Dopo l'ispezione scopriamo:

```
<div class="expander-icon-wrapper spoiler-warning control">  
  <svg class="ipl-expander__icon expander-icon "  
width="12" height="8" viewBox="0 0 12 8"  
xmlns="http://www.w3.org/2000/svg">
```

By. CLASS

Spoiler

```
expanders = driver.find_elements(By.CLASS,  
    "spoiler-warning__control") # 41 spoiler  
for expander in expanders:  
    if expander.is_displayed():  
        expander.click()  
        time.sleep(1) # no d.o.s. but also wait for data
```


BeautifulSoup

- Ora il browser Chrome ha tutte le reviews caricate
- Usiamo l'attributo `page_content` per caricare l'intero HTML
- Sempre con l'ispezione notiamo che tutte le reviews sono contenute in:

```
<div class="lister-item mode-detail imdb-user-review  
collapsible" data-review-id="rw2556823" data-vote-  
url="/title/tt0042332/review/rw2556823/vote/interesting" data-  
initialized="true">
```

...

```
</div>
```

- Analizziamo l'HTML con BeautifulSoup

BeautifulSoup

di un close()



```
html = driver.page_source # riceve il sorgente da Selenium
soup = BeautifulSoup(html, "html.parser") # crea il soup
# cerca tutti i div delle reviews
divs = soup.find_all("div", {"class": "lister-item"})
```

Review

- Ogni singola review può avere:
 - rating (``)
 - titolo (``)
 - autore (``)
 - data (``)
 - testo (`<div class="text show-more__control clickable">`)
- Andiamoli ad estrarre

BeautifulSoup

```
title = divs[0].find("a", {"class": "title"})
if title is not None:
    print(title.text.strip())
rating = divs[0].find("span", {"class": "rating-other-user-rating"})
if rating is not None:
    print(rating.text.strip())
author = divs[0].find("span", {"class": "display-name-link"})
if author is not None:
    print(author.text.strip())
rev_date = divs[0].find("span", {"class": "review-date"})
if rev_date is not None:
    print(rev_date.text.strip())
review = divs[0].find("div", {"class": "text"})
if review is not None:
    print(review.text.strip())
```

Database

- Ora salviamo tutte le reviews in un database *sqlite3*
 - Creiamo una tabella con gli attributi:
 - `idfilm` (text: codice imdb vedi url della pagina)
 - `idreview` (text: codice imdb della review)
 - `title` (text)
 - `reviewdate` (text)
 - `rating` (int)
 - `text` (text)
 - `author` (text)
- ```
CREATE TABLE reviews (
 idfilm TEXT,
 idreview TEXT,
 author TEXT,
 title TEXT,
 reviewdate TEXT,
 rating INT,
 text TEXT
);
```

# Sqlite

---

```
query = """
CREATE TABLE IF NOT EXISTS reviews (
 idfilm TEXT,
 idreview TEXT,
 author TEXT,
 title TEXT,
 reviewdate TEXT,
 rating INT,
 text TEXT
);"""
apre il db o lo crea
conn = sqlite3.connect("imdb.db")
se la tabella non esiste la crea
conn.execute(query)
salva su disc
conn.commit()
```

# Sqlite e dict

---

- Con qualche trucco si possono usare i *dict* per popolare un *db*
  1. Costruire un *dict* con chiavi uguali agli attributi e valori ai campi
  2. Scrivere una funzione che crei la query e la lista dei valori:

# Sqlite e dict

```
def create_and_execute_query(conn, row):
 # elenco dei nomi campi separati da ,
 columns = ', '.join(row.keys())
 # elenco di ? separati da virgola
 placeholders = ', '.join('?' * len(row))
 # l'operatore {} è simile al placeholder %s
 # il metodo format sostituisce al primo {} la stringa columns
 # e al secondo {} la stringa dei placeholders
 sql = 'INSERT INTO reviews ({{}}) VALUES ({{}})'.format(columns,
 placeholders)
 # values è un vettore di valori
 values = [x for x in row.values()]
 # execute con due valori sostituisce tutti i ?
 # presenti nella query sql
 # con i corrispettivi valori nella lista values
 conn.execute(sql, values)
 # scrivo su disco
 conn.commit()
```

*[ [1, 2, 3], [4, 5, 6] ]*

*? , ? , ?*



# Salvataggio dati

- Mettiamo insieme il tutto:

```
FILM_CODE = "tt0042332" # dalla url
```

```
for div in divs:
 row = {}
 row["idfilm"] = FILM_CODE
 title = div.find("a", {"class": "title"})
 row["title"] = None if title is None else title.text.strip()
 rating = div.find("span", {"class": "rating-other-user-rating"})
 row["rating"] = (None if rating is None
 else int(rating.text.strip().split("/")[0]))
 author = div.find("span", {"class": "display-name-link"})
 row["author"] = None if title is None else author.text.strip()
 rev_date = div.find("span", {"class": "review-date"})
 row["reviewdate"] = (None if title is None
 else rev_date.text.strip())
 review = div.find("div", {"class": "text"})
 row["text"] = None if title is None else review.text.strip()
 create_and_execute_query(conn, row)
```