# Selenium

Selenium è un framework di test sviluppato in Java che permette di interagire con diversi browser Web. In effetti permette di realizzare dei test dinamici che facilitano dei test funzionali e quindi non richiedono l’esecuzione di un software. Per realizzare i nostri test abbiamo utilizzato l’estensione che si chiama Selenium IDE che può essere installata su qualsiasi browser.

## Librerie o package usate pero lo scrapper

Re: libreria per le espressioni regolari

Time: libreria che contiene dei metodi per il tempo.

Datetime: libreria che contiene dei metodi per le date.

Selenium: Package di selenium che contiene dverse librerie.

Webdriver: Libreria che permette di determinare il browser da utilizzare per realizzare i test.

Options: Libreria che contiene diverse opzioni per webdriver.

Bs4: Package che contiene diversi metodi della libreria beautiful soup.

Beautiful soup: Libreria che permette di parsare del codice HTML.

## Estrapolazione dati (Scrapper)

Per estrapolare i dati dal sito abbiamo creato uno script. Per realizzarlo ci siamo basti su quello del professore Petrini, però modificando alcune cose. Il browser che utilizziamo per eseguire i test è chrome, quindi nello script abbiamo dovuto specificare quale driver utilizzare. Per farlo abbiamo solo dovuto scrivere questa riga di codice:

self.browser = webdriver.Chrome()

Ogni volta che testiamo questo scrapper apre il browser Google Chrome andando sul sito dell'orario della scuola ed inseguito esegue i vari test. Questo possiamo solo lasciarlo in fase di sviluppo, ma certamente per il sistema finale (“GUI”) dobbiamo nasconderla, per farlo dobbiamo aggiungere le options come parametro in più nel webdriver.Chrome( ).

opts = Options()

opts.set\_headless()

assert opts.headless

self.browser = webdriver.Chrome(options=opts)

Per raggiungere un sito web dobbiamo fare una richiesta GET con webdriver di selenium.

self.browser.get(self.url)

Questo comando permette di verificare che un determinato elemento sia caricato. All’interno del metodo loaded\_check bisogna mettere l’id del elemento che disederiamo caricare. In questo caso l’id corrisponde al link interno “corsi”.

self.loaded\_check("GInterface.Instances[0].Instances[0]\_Combo1")

Per potere ricavare un elemento HTML con selenium bisogna utilizzare il metodo: find\_element\_by\_id() che permette di trovare un elemento con un determinato id.

Per cliccare diversi elementi di un sito con selenium bisogna utilizzare il metodo click().

Ad esempio questa linea di codice permette di cliccare il link interno “corsi”.

self.browser.find\_element\_by\_id("GInterface.Instances[0].Instances[0]\_Combo2").click()

Per potere scrivere il nome della classe all’interno della barra della ricerca abbiamo utilizzato il seguente codice:

Permette di selezionare la barra di ricerca che permette di cercare l’orario di una classe.

cerca\_orario = self.browser.find\_element\_by\_id("GInterface.Instances[1].Instances[1].bouton\_Edit")

Il metodo clear() permette di cancellare il contenuto che c’è all’interno della barra di ricerca (magari in precedenti ricerca nella barra di ricerca c’è scritto qualcosa che dobbiamo per forza cancellare).

cerca\_orario.clear()

Il metodo send\_keys() permette di scrivere all’interno della barra di ricerca.

cerca\_orario.send\_keys(ricerca)

Permette di cliccare la “lente” che avvia la ricerca.

self.browser.find\_element\_by\_id("GInterface.Instances[1].Instances[1].bouton\_Bouton").click()

Durante la ricerca di una classe possiamo anche definire se ricavare l’oriario della settima successiva a quella corrente. Per farlo abbiamo realizzato nello scrapper un metodo che si chiama clicca\_numero\_settimana\_dopo( ). Ogni volta che vogliamo cercare possiamo impostare al metodo cerca\_orario\_aule( ) un parametro a true o a false. Quest’ultimo ci permette di attivare un metodo (clicca\_numero\_settimana\_dopo) che è quello che andrà a cliccare in cima il numero della settimana successiva.

Per potere cambiare la visualizzazione dei dati dell’orario nella modalità elenco abbiamo aggiunto queste linee. Nel codice abbiamo aggiunto uno sleep, questo perché aétrimenti non riusciva ad eseguire l’animazione del menu a tendina che scende.

# apre il menu per cambiare la visulalizzazione

self.browser.find\_element\_by\_id("GInterface.Instances[0].Instances[2].bouton\_Bouton").click()

time.sleep(1)

# clicca su in elenco

self.browser.find\_element\_by\_id('GInterface.Instances[0].Instances[2]\_1').click()

time.sleep(1)

Una volta che abbiamo scelto la classe dovevamo selezionare la tabella dell’orario e iniziare a parsare i dati. Per farlo abbiamo aggiunto queste linee di codice dove utilizziamo la libreria beautiful soup.

soup = BeautifulSoup(self.browser.page\_source, features="html5lib")

# cerca la tabella che contiene l'orario

table = soup.find('table', id="GInterface.Instances[1].Instances[8]\_Contenu\_0")

All’interno del metodo BautifulSoup dobbiamo definire su quale pagina web vogliamo parsare i dati. Con il metodo find possiamo andare a cercare nella pagina web un determinato tag con un determinato id.

Prima di parsare i dati controlliamo se la tabella cercato contiene dei dati, questo perché potrebbe essere una settimana di vacanza dove non ci sono lezioni. Inseguito per potere selezionare gli elementi della tabella dobbiamo cercare tutti tag tr che contengono dei tag td o th. Per farlo abbiamo aggiunto la seguente linea di codice.

rows = table.find\_all('tr', recursive=True)

Con il metodo find\_all possiamo catturare un tag che presente più volte all’interno della pagina web. Inseguito abbiamo pensato anche in quale modo salvare le varie informazioni all’interno di un file JSON. Nella immagine potete vedere la struttura del file JSON che abbiamo adottato.

Immagine che contiene screenshot

Descrizione generata automaticamente

Per realizzare questa struttura abbiamo creato le seguenti variabili.

settimana = [] # Contiene tutto l'orario della settimana

giorno = {'giorno': None, 'materie': None} # Contiene il giorno in forma testuale e le materie del giorno

materie = [] # Contiene le materie del giorno

lezione = {'time': None, 'lesson': None, 'teacher': None, 'classe': None} # Contiene i dati della lezione.

primaVolta = True # Serve ha non far fare l'inserimento nei dizionari al primo ciclo

Inseguito abbiamo realizzato il seguente codice:

for row in rows:

tds = row.find\_all('td', recursive=True)

for i, td in enumerate(tds):

if td.has\_attr('class') and td['class'][0] == 'Gras':

if (not primaVolta):

giorno['materie'] = materie

materie = []

settimana.append(giorno)

giorno = {'giorno': None, 'materie': None}

primaVolta = False

giorno['giorno'] = td.text.strip()

elif td.has\_attr('style'):

if (i == 2):

lezione['time'] = td.text.strip()

# Cerca Lezione

if (i == 3):

lezione['lesson'] = td.text.strip()

# Cerca docente

if (i == 4):

lezione['teacher'] = td.text.strip()

# Cerca Classe

if (i == 5):

lezione['classe'] = self.parse\_classe(td.text)

materie.append(lezione)

lezione = {'time': None, 'lesson': None, 'teacher': None, 'classe': None}

break

giorno['materie'] = materie

settimana.append(giorno)

return settimana

1. Il primo ciclo corrisponde ad ogni elemento tr della tabella
2. Il secondo ciclo corrisponde ad ogni elemento td della tabella
3. Al primo ciclo dobbiamo inserire all’interno dell’array settimana il giorno della settimana.
4. Invece per i cicli successivi fino al giorno successivo si deve inserire in ogni dizionario i dati.

Alcuni metodi utilizzati per questo codice sono i seguenti:

* has\_attr 🡺 permette selezionare un tag con un determinato attributo (class, id, style, …)
* text.Strip() 🡺 permette di selezionare il contenuto di un tag.
* append 🡺 permette aggiungere degli elementi ad un array.
* parse\_classe 🡺 metodo creato da noi per potere formattare nel modo corretto la classe. All’interno di questo codice abbiamo realizzato un espressione regolare che corrisponde al testo che vogliamo scrivere all’interno del dizionario classe.

## Flask

Flask è un mini-framework in Python che viene utilizzato per lo sviluppo web e permette di creare dei propri siti web in modo dinamico e interattivo.

Abbiamo creato un file che si chiama app.py che sarà l’applicazione flask. All’interno per difinire che si tratta di un flask dobbiamo importare la libreria flask e aggiungere la seguente linea di codice.

app = Flask(\_\_name\_\_)

Inseguito bisogna difinire la app routing che viene utilizzato per mappare l’URL specifico con la funzione associata che intende svolgere alcune attività (struttura simile al MVC) .

Nel nostro caso l’URL è associato alla funzione aule che permette di stampare l’orario di una determinata classe. Sì può anche aggiungere come parametro il tipo di richiesta che si vuole fare. In questo caso utilizziamo il metodo GET che consiste nell’accordare all’indirzzo della pagina web i diversi parametri contenenti i dati che si vogliono trasmettere. In questo caso vogliamo trasmettere se mostrare i dati della settimana corrente o quella successiva.

@app.route('/aule', methods=['GET'])

Per ricavare i parametri che sono presenti nel URL bisogna usare il metodo request.args.get.