```
9
10  # 브라우저 꺼짐 방지 + 자동화 메세지 제거 옵션
11  chrome_options = Options()
12  chrome_options.add_experimental_option("detach", True)
13  chrome_options.add_experimental_option("excludeSwitches", ["enable-automation"])
14
15  # 브라우저(크롬) 실행
16  driver = webdriver.Chrome(options=chrome_options)
17
18  # 주소 입력
19  url = "https://www.melon.com/genre/song_list.htm?gnrCode=GN0500#params%5BgnrCode%5D=GN0500&params%5BdtlGnrCode%5D=GN0501&params%5BorderBy%5
20  driver.get(url)
21
22  # 페이지 로딩
23  time.sleep(2)
24
25  # 뮤지션 정보를 가져온다
26  musician = driver.find_elements(By.CSS_SELECTOR, '#songList > form#frm > div.service_list_song > table > tbody > tr')
27
28  # 뮤지션 리스트 생성
29  musician_list = []
30
```

웹 페이지를 열기 위한 코드

크롤링 하려는 페이지의 url 입력

데이터를 크롤링 하기 위해 해당하는 CSS_SELECTOR

(HTML의 태그 혹은 class, id 값을 참조)

```python
# 뮤지션 리스트 생성
musician_list = []

for singer in musician:
    # 가수 ID 추출
    singer_info = singer.find_element(By.CSS_SELECTOR, 'div.ellipsis.rank02 > a')
    onclick_attribute = singer_info.get_attribute('href')

    # ID 추출 (예: 'javascript:melon.link.goSongDetail('38179751');'에서 38179751만 추출)
    singer_id = onclick_attribute.split("'")[1]

    # 고유 ID로 곡 URL 생성
    musician_url = f"https://www.melon.com/artist/detail.htm?artistId={singer_id}"
    print(musician_url)

    musicians_url = {
        "musician_url": musician_url
    }

    musician_list.append(musicians_url)
```

뮤지션들을 담을 빈 리스트 생성

뮤지션들의 고유 ID를 추출

(추출하는 이유 : 해당하는 ID를 보고

웹 페이지를 이동하며 크롤링 하기 위해)

고유 ID를 이용해 뮤지션 상세 페이지에 대한 url 생성

```python
# 상세 정보 리스트
detail_list = []

def musician_detail(detail):
    # 음악 상세 페이지로 이동
    driver.get(detail['musician_url'])
    time.sleep(1)

    # 변수 초기화
    img_src = None
    artist_name = None
    artist_name = None
    debut_value = None
    birthday_value = None
    activity_type_value = None
    agency_value = None
```

크롤링이 끝날 때 마다 뮤지션 상세 페이지로 이동

디테일을 담기 위한 빈 리스트 생성

데이터 수집 후 새로운 크롤링을 하기 위해 변수를 초기화

```python
# 상세 정보 크롤링
artist_detail_info = {}
```

```python
# 이미지 크롤링
try:
    img = driver.find_element(By.CSS_SELECTOR, 'div.wrap_dtl_atist > div.dtl_atist.clfix > div.wrap_thumb > span.thumb > span#artistImgArea > ir
    img_src = img.get_attribute('src')
    artist_detail_info["이미지"] = img_src
except NoSuchElementException:
    artist_detail_info["데뷔"] = None
```

```python
# 아티스트 이름 크롤링
try:
    artist_name = driver.find_element(By.CSS_SELECTOR, 'div.wrap_atist_info > p.title_atist').text
    artist_detail_info["이름"] = artist_name
except NoSuchElementException:
    artist_detail_info["이름"] = None
```

```python
try:
    # 데뷔 정보
    debut_dt = driver.find_element(By.XPATH, "//dl[@class='atist_info clfix']/dt[text()='데뷔']")
    debut_value = debut_dt.find_element(By.XPATH, "following-sibling::dd[1]/span[@class='gubun']").text
    artist_detail_info["데뷔"] = debut_value  # "데뷔"와 그에 대한 값
except NoSuchElementException:
    artist_detail_info["데뷔"] = None

try:
    # 생일 정보
    birthday_dt = driver.find_element(By.XPATH, "//dl[@class='atist_info clfix']/dt[text()='생일']")
    birthday_value = birthday_dt.find_element(By.XPATH, "following-sibling::dd[1]").text
    artist_detail_info["생일"] = birthday_value  # "생일"과 그에 대한 값
except NoSuchElementException:
    artist_detail_info["생일"] = None
```

크롤링을 할 때 데이터가 있다면 크롤링을 시도, 없을 수도 있기 때문에 예외처리를 사용해 주었다.

크롤링 시 CSS_SELECTOR 뿐만 아니라 XPATH를 사용해 크롤링도 가능

```python
# 랜덤 고유 ID 생성 함수
def generate_unique_id(length=8):
    # 알파벳 대소문자 + 숫자
    characters = string.ascii_letters + string.digits
    return ''.join(random.choice(characters) for _ in range(length))

# 고유 ID 생성
unique_id = generate_unique_id()

# 상세 정보 딕셔너리
detail_info = {
    "musician_ID": unique_id,
    "img_src": img_src,
    "artist_name": artist_name,
    "debut_value" : debut_value,
    "birthday_value" : birthday_value,
    "artivity_type" : activity_type_value,
    "agency" : agency_value
}

# detail_list에 상세 정보 추가
detail_list.append(detail_info)
```

각 뮤지션 마다 고유 ID를 알파벳 대, 소문자와 숫자 0~9까지의 랜덤한 문자열을 생성

크롤링한 데이터를 딕셔너리 생성

생성한 딕셔너리를 디테일 리스트에 추가

```python
# 리스트를 DataFrame으로 변환
df = pd.DataFrame(detail_list)

# # 엑셀 파일로 저장
# df.to_excel("music_data.xlsx", index=False)

# CSV 파일로 저장
df.to_csv("musician_data10.csv", index=False, encoding="utf-8-sig")

print("파일 저장 완료")
```

디테일 리스트를 DataFrame로 변환

(DataFrame는 pandas 라이브러리에서 제공하는 2차원 데이터 구조로
행과 열로 구성된 테이블 형태의 데이터를 처리할 수 있다.)

변환한 데이터를 CSV 또는 엑셀 파일로 저장

크롤링이 끝났다는 확인 문구