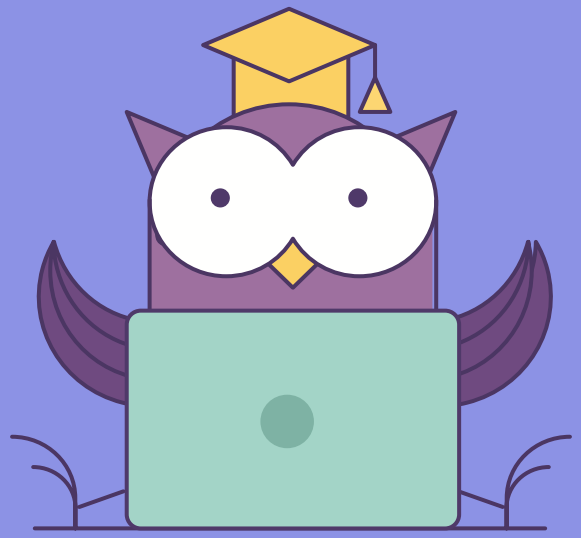
ОНЛАЙН-ОБРАЗОВАНИЕ

Меня хорошо слышно && видно?



Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

Не забыть включить запись!

Протоколирование и отчетность

Python QA Engineer

Правила вебинара

Паузы между блоками

Вопросы в чат

slack

Обсуждения в Slack

Цели вебинара

1. Логирование
2. Библиотека logging
3. AbstractEventListener
4. Логи браузера
5. Логирование proxy

01

logging

logging

Логирование - это способ сохранения информации о работе системы или программы. Как правило логи (артефакты логирования) используются для анализа работы или поиска ошибок и неисправностей.

Чаще всего, при локальной разработке сподручней использовать банальный print() или дебаггер.

Однако, если возникает необходимость анализа сложной системы, или к коду которой прямо сейчас нет прямого доступа и анализировать необходимо в рантайме, то без логов никуда.

Python предоставляет очень качественный модуль для логирование logging как часть стандартной библиотеки.

logging

Построение качественной системы логирования задача сложная

1) Что именно писать в логи
2) Хранение логов
3) Доступность логов
4) Понятность и достаточность
5) Соответствие архитектуре

logging

Выделяют 5 основных уровней логирования и в общепринятой практике каждый и решает определенную задачу, отвечает за определенный класс сообщений от системы.

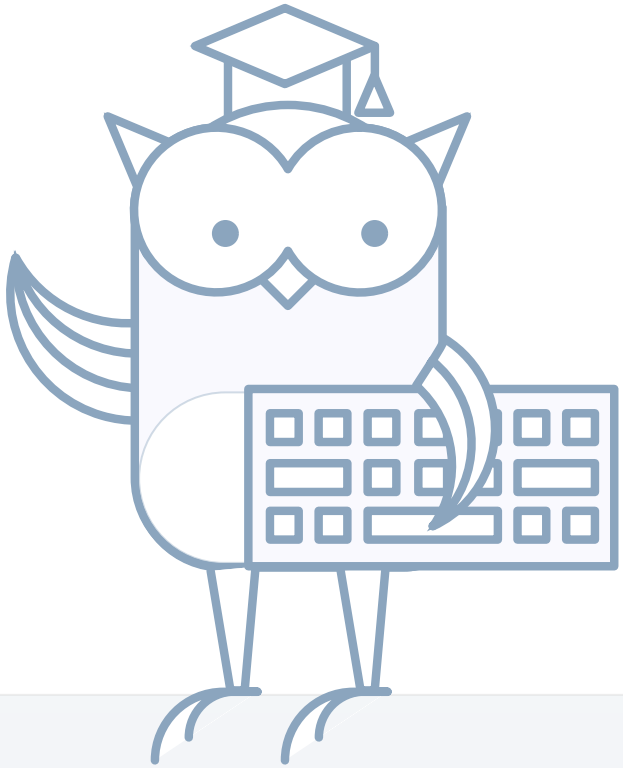
- DEBUG - Самый полный уровень, обычно используют при разработке или поиске проблем в системе.
- **INFO** - Любая информация сообщающая о выполнении системой каких-либо операций или действий.
- **WARNING** - Информация цель которой сообщить или предупредить пользователя о какой-то потенциальной проблеме (место на диске подходит к концу).
- **ERROR** - Серьезная ошибка в системе, которая не позволяет ей выполнить одну из важных функций или операций.
- **CRITICAL** - Ошибка из-за которой система не может продолжать функционировать.

logging

1) Создание простого логгера
2) Базовый конфиг для логгера
3) Логирование вложенных модулей
4) Название логгера по имени модуля
5) Настройка формата сообщений
6) Loggers, Handlers, Formatters

Ссылки:
https://docs.python.org/3/howto/logging.html

Вопросы



02

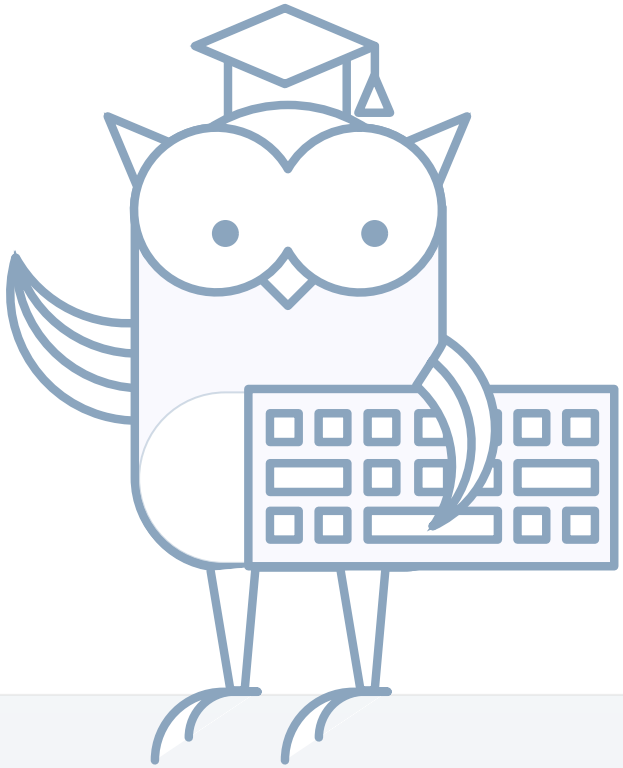
AbstractEventListener

event listeners

Селениум предоставляет возможность переопределить поведение при действиях:

```
#after_change_value_of(element, driver) ⇒ Object
#after_click(element, driver) ⇒ Object
#after_close(driver) ⇒ Object
#after_execute_script(script, driver) ⇒ Object
#after_find(by, what, driver) ⇒ Object
#after_navigate_back(driver) ⇒ Object
#after_navigate_forward(driver) ⇒ Object
#after_navigate_to(url, driver) ⇒ Object
#after_quit(driver) ⇒ Object
#before_change_value_of(element, driver) ⇒ Object
#before_click(element, driver) ⇒ Object
#before_close(driver) ⇒ Object
#before_execute_script(script, driver) ⇒ Object
#before_find(by, what, driver) ⇒ Object
#before_navigate_back(driver) ⇒ Object
#before_navigate_forward(driver) ⇒ Object
#before_navigate_to(url, driver) ⇒ Object
#before_quit(driver) ⇒ Object
```

Вопросы



03

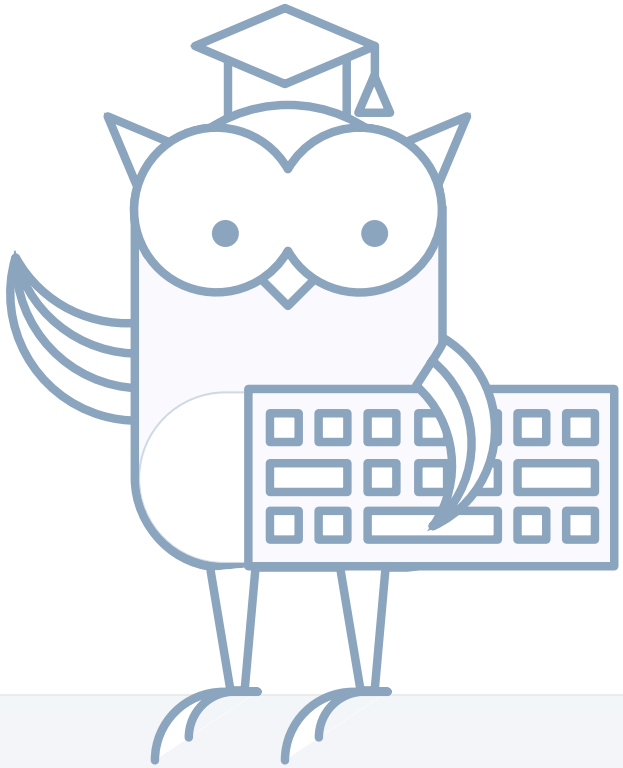
browser logs

browser and performance logs

https://github.com/SeleniumHQ/selenium/wiki/Logging

Пример:
https://github.com/konflic/python_qa_logging/blob/master/3_browser_driver_performance/browser_log.py

Вопросы



Опрос

https://otus.ru/polls/10233/

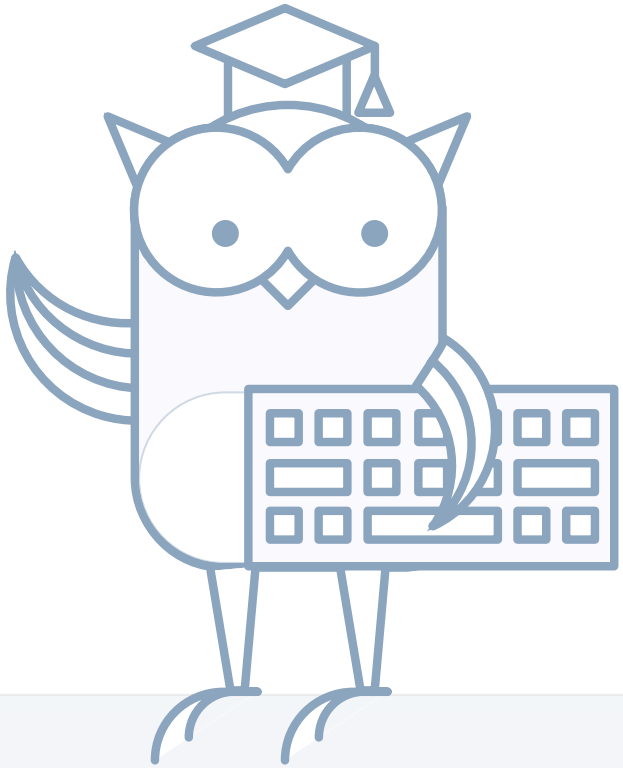
04

Proxy with selenium

proxy

https://github.com/lightbody/browsermob-proxy/

Вопросы




gifak.net